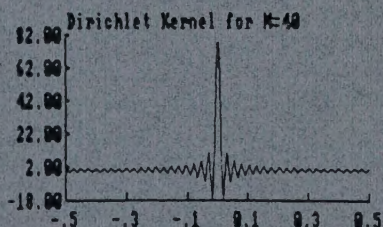
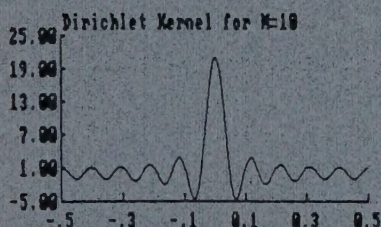
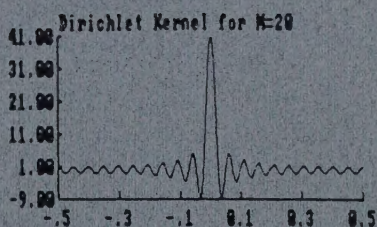
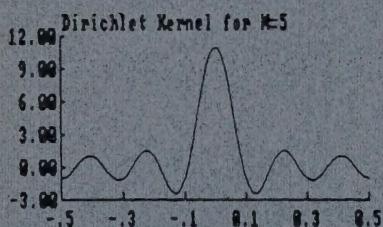


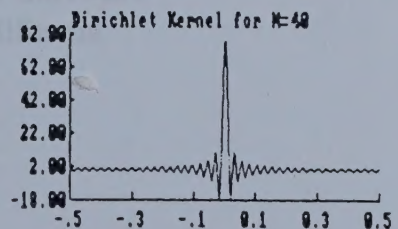
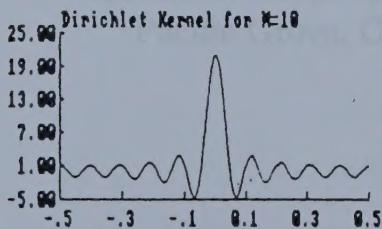
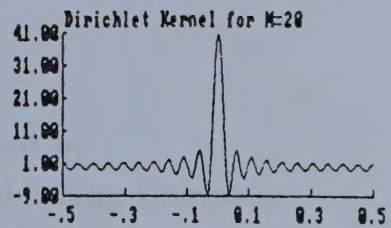
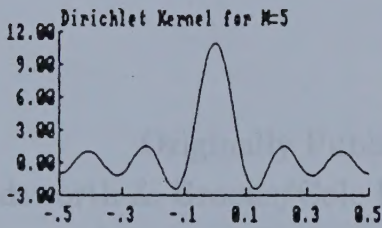
TIMESLAB: A Time Series Analysis Laboratory

H. Joseph Newton
Texas A&M University



TIMESLAB: A Time Series Analysis Laboratory

H. Joseph Newton
Texas A&M University



TIMESLAB:
A TIME SERIES ANALYSIS LABORATORY

TIMESLAB: A Time Series Analysis Laboratory

H. Joseph Newton
Texas A&M University

**Originally Published by
Wadsworth & Brooks/Cole Publishing Company
Advanced Books and Software
Pacific Grove, California**

*To My Parents, My Wife, Linda,
and My Children, Karah and Timothy*

© 1996 by H.J. Newton. All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior written permission of H.J. Newton, Department of Statistics, Texas A&M University, College Station, Texas 77843. This is a reprint of the version of this book published originally by Wadsworth & Brooks/Cole in 1988.

Library of Congress Cataloging-in-Publication Data

Newton, H. Joseph, [date]

TIMESLAB: a time series analysis laboratory.

Bibliography; p.

Includes indexes.

1. TIMESLAB (Computer program) 2. Time-series analysis—Data procession. I. Title.
QA280.N48 ISBN 0-534-09198-9

PREFACE

This book and its accompanying software are my attempt to bring together in one package the basic theory, methods, and algorithms that are used to analyze univariate and bivariate linear time series in both the time and frequency domains. I have tried to be complete and rigorous in my treatment of each of the three areas, but at the same time I have tried to organize the material in such a way that makes it possible for readers particularly interested in one area to find their way around in the package without too much distraction. My main aim has been to provide a set of tools that allows a reader to study the various parts of the subject. I have called the result a laboratory as its emphasis is not primarily on providing “canned” programs for doing traditional analyses, but rather on allowing the interested reader to study the theory and methods involved in the subject. A set of routines (called macros) is in fact included that allows TIMESLAB to be used as a canned program, but the reader is encouraged to study the subject in more detail.

TIMESLAB grew out of a series of lecture notes and computer programs that I developed over ten years in a course in time series analysis for students having varied interests and backgrounds in mathematics and statistics. These students have included (1) business students primarily interested in forecasting, (2) engineers wanting to learn spectral analysis, (3) students in statistics wanting to learn general theory and methods, and (4) students whose primary interest was in the computational aspects of the subject. All of these students have had in common a familiarity with the basic ideas of probability and distributions, a knowledge of regression analysis from a matrix point of view, and the desire to learn about time series analysis. The software that I developed originally took the form of both a library of Fortran subroutines and a collection of batch style main programs. Both of these were difficult to use and expensive to run. As personal computers became available in the early 1980s, it became clear that they made possible the creation and widespread use of command-driven interactive graphics style programs for studying a wide variety of scientific subjects, including time series analysis.

The Program

The TIMESLAB program consists of approximately 150 "commands," each consisting of an operation such as reading, plotting, transforming, or forecasting data. Several graphics commands are available (all of the figures included in the book were created using the program), a text editor is built in, and it is possible to issue operating system commands while inside the program. The user tells the program what to do either by entering commands one at a time from the keyboard or by issuing a command telling TIMESLAB to execute a series of commands contained in a file. Enough conditional and unconditional branching commands are provided for use in these (macro) files so that TIMESLAB is actually a time series analysis programming language. Thus the commands can be thought of as a library of subroutines, while the macros can be thought of as main programs. The users can use the 80 macros discussed in the book and provided with the program, or else create their own macros for performing analyses not covered by the supplied macros. An appendix (Appendix B) is included that guides the user through the process of creating macros, and the code of each of the supplied macros is listed in the book to provide models for the writing of others.

In addition to commands directly associated with time series analysis, there are several general-purpose commands, including matrix operations, evaluating the pdf, cdf, and quantile functions of the Z , t , χ^2 , and F distributions, a general nonparametric probability density estimation procedure, and several commands for analyzing polynomials. This has led to its use in a variety of non-time series analysis applications.

The Textbook

I have tried to provide a balance of theory, methods, and algorithms in the book. The theory is stated rigorously but in a manner in which the reader interested only in applications can skip over the details. The basic results for each major topic are gathered into one (often lengthy) theorem. At the end of many of the theorems I have included a section describing the implications of the theorem. This approach makes it possible for readers to go directly to the material of interest to them without having to search through a series of definitions, theorems, and proofs. Some proofs are included if they are not extremely difficult and not widely available elsewhere. If TIMESLAB were to be used in a course principally devoted to theory, the instructor would have to supplement the proofs included in the book. References to places where proofs can be found are included in the cases where they are not given. An index to the theorems is included in Appendix E, while Appendix A contains a discussion of the more difficult mathematical and statistical background needed for those interested in the theory. In the one-semester course described above, I have emphasized understanding of these implications more than the rigorous statements themselves.

I have taken the same approach with the algorithms that are used in the TIMESLAB program. For example, the fast Fourier transform is described briefly in Chapter 1 of the book and more extensively in Appendix A. Many of the algorithms for univariate time series are carefully explained, but usually in separate sections. For example, Section 2.6 is devoted to a description of many of the basic algorithms for univariate time series.

For those readers primarily interested in the application of time series methods, several of the macros can be used as models for similar analyses. Appendix E contains a list of all of the examples and macros that are included in the book.

The basic point of view taken by the book is that much is to be gained both in theory and in applications by combining what are called the time domain and frequency domain approaches to time series analysis. The emphasis is on the use of the correlogram, partial correlogram, and spectral density function for both model determination and as diagnostic tools. Many of the ideas of time series analysis are presented in analogy with ideas from regression analysis. A section is included in Appendix A reviewing these ideas. Thus the usual modeling strategy of regression analysis is employed with a wide range of models and diagnostics, including those from the frequency domain, being easily accessible in the TIMESLAB program.

The book is organized into four chapters, a mathematical and statistical appendix, two appendices describing the TIMESLAB program, an appendix containing time series data sets, and finally an appendix that is an index to the commands, theorems, macros, examples, and figures that are in the book. The first three chapters are concerned with univariate time series, while the fourth discusses bivariate series. In Chapter 1 the ideas of the time and frequency domains are introduced in terms of describing and classifying time series data sets. The sample correlogram, partial correlogram, and spectral density are introduced as tools for making an exploratory analysis of data. This exploration is presented by determining whether transformations of the data are required. Traditional data transformations are then discussed and some simple forecasting methods presented. Chapter 1 concludes with a discussion of some basic methods for describing the distribution of a data set. This is useful for investigating assumptions made about data and also as a tool to be used later for some numerical experiments illustrating theory.

Chapter 2 presents the basic probability theory of covariance stationary time series, with a discussion of the traditional models used in time series, and the basic theory of spectral analysis and minimum mean square error prediction. Chapter 3 consists of a description of a variety of statistical inferences that can be made from data. The sampling properties of the descriptive statistics introduced in Chapter 1 are given, two tests for white noise are described, and traditional nonparametric spectral estimation is presented in some detail. In Sections 3.4 and 3.5, the modern methods of parameter estimation and model order determination are described. In Section 3.6, the popular Box-Jenkins modeling and forecasting method is described, while Section 3.7 discusses other

such model-based procedures, including regression with autocorrelated errors and iterated autoregression modeling. After all of these modeling techniques are developed in previous sections, Sections 3.8 and 3.9 describe their use for spectral density estimation and the search for periodicities, respectively.

The analysis of bivariate time series is described in Chapter 4 with the emphasis being on the ideas new to that situation. These ideas include cross-spectral analysis and transfer function modeling.

At the end of each of Chapters 1-4, there are a series of examples that either are used in the chapter or supplement the ideas of the chapter. There are also two sets of problems at the end of each chapter: one set requiring the reader to use the program to illustrate a theoretical or methodological point, and the other set not requiring the program.

Suggestions for Using TIMESLAB

TIMESLAB has been designed to be used in a variety of types of courses. For a two-semester sequence for students with a good background in mathematics and statistics, the entire book could be covered, with Appendix A, Chapters 1 and 2, and the first six sections of Chapter 3 covered in the first semester and the rest in the second semester. The sections on computing aspects (Sections 2.6 and A.1 and A.3) could be optional. I would assign about one third to one half of the computational and theoretical problems from each chapter.

A one-semester course in applications of univariate time series analysis could consist of Sections 1-6 of Chapter 1, a rapid treatment of Sections 2.1, 2.2, and 2.5, and then an introduction to each of the topics in Chapter 3. Approximately one fourth of the course would be devoted to each of the first two chapters and half to the third. The emphasis would be on the implication of the use of the theorems, and the supplied macros and examples would be used to make it possible to cover all of the material.

Acknowledgements

I learned time series analysis from Emanuel Parzen and my heartfelt thanks go to him for all of his guidance and support since 1973. He has always been a strong advocate of the combining of the time and frequency domain approaches to time series analysis and the need for flexible software for studying all aspects of the subject. I hope that TIMESLAB will help foster this view. Special thanks are also offered to Marcello Pagano for his guidance and collaboration and to Richard Schmidt for first showing me the joys of computing.

To my colleagues at Texas A&M I owe a great personal debt for the constant encouragement they have given me and the stimulating atmosphere they have provided.

Several people have helped in the creation of TIMESLAB while students at

Texas A&M. The program is written in a combination of fortran and assembly language and I typed the book using the \TeX typesetting language. Avi Harpaz and Will Alexander helped me learn IBM assembly language and made many important suggestions about the structure and content of the program. Staci Sontag showed me how to design the fortran data structures for a program such as TIMESLAB and Anne Coleman helped in some aspects of the design of the \TeX macros that I used in typing the book. Several of the students in my time series classes made important suggestions about the book and program.

Very special thanks are offered to the reviewers of TIMESLAB during its preparation and in particular to David A. Burn who made many important suggestions about the structure and format of both the book and the program.

I would particularly like to thank the Office of Naval Research for their support for many years; most recently under Contract N00014-84-K-0350. This support has established an environment that has made it possible to bring together the ideas and computational aspects of time series analysis into TIMESLAB.

Finally, I would like to thank my wife Linda for her support and understanding during the creation of TIMESLAB. I could not have done it without her.

H. JOSEPH NEWTON

College Station, Texas

December, 1987

CONTENTS

Chapter 1	
Describing, Transforming, and Forecasting Univariate Time Series	1
1.1. Time Series Data Types	1
1.2. Time Series Memory Types	3
1.3. Aims of Time Series Analysis	5
1.4. Basic Descriptive Statistics	6
1.4.1. The Sample Correlogram	6
1.4.2. The Sample Partial Correlogram	10
1.4.3. The Periodogram and Sample Spectral Density	11
1.4.4. The Relationship of the Correlogram and Periodogram	22
1.5. Transformations	29
1.5.1. Stabilizing Variance	30
1.5.2. Removing Trends	30
1.5.3. Accounting for Seasonal Variability	32
1.5.4. General Smoothing Operations	33
1.6. Some Simple Forecasting Methods	35
1.6.1. The Inverse Differencing Method	35
1.6.2. The Regression Method	36
1.6.3. Simple Moving Average	37
1.6.4. Simple Exponential Smoothing	38
1.6.5. Difference Equations and the Behavior of Forecasts	38
1.7. Describing the Distribution of Data	42
1.8. Examples and Problems	46
Chapter 2	
Probability Theory for Univariate Time Series	67
2.1. Introduction	68

2.2.	Covariance Stationary Time Series	70
2.3.	The Theory of Linear Filters	78
2.4.	Time Series Prediction	86
2.5.	Time Series Models	90
	2.5.1. Random Walk Processes	90
	2.5.2. Harmonic Processes	92
	2.5.3. Moving Average Processes	94
	2.5.4. Autoregressive Processes	98
	2.5.5. Autoregressive-Moving Average Processes	107
	2.5.6. Subset and Multiplicative Subset ARMA Processes	112
	2.5.7. ARIMA Processes	113
	2.5.8. ARMA Models and Inverse Autocovariances	115
	2.5.9. The General Linear Process	117
	2.5.10. Nonlinear and Non-Gaussian Processes	118
2.6.	Algorithms for Univariate Time Series	118
	2.6.1. Prediction Using the Cholesky Decomposition	118
	2.6.2. ARMA Prediction and the Kalman Filter Algorithm	131
	2.6.3. The Cramer-Wold Factorization	134
	2.6.4. Factoring an ARMA Covariance Generating Function	140
2.7.	Examples and Problems	143

Chapter 3

Statistical Inference for Univariate Time Series 158

3.1.	Sampling Properties of Descriptive Statistics	160
	3.1.1. The Sample Mean	161
	3.1.2. The Sample Autocovariances and Autocorrelations	163
	3.1.3. The Sample Spectral Density Function	168
3.2.	Tests for White Noise	172
	3.2.1. Bartlett's Test	172
	3.2.2. The Portmanteau Test	174
3.3.	Nonparametric Spectral Density Estimation	176
	3.3.1. Smoothing the Sample Spectral Density	177
	3.3.2. Kernels and Fourier Series Approximations	180
	3.3.3. Sampling Properties of Estimators	188
3.4.	Estimating the Parameters of ARMA Models	194
	3.4.1. Maximum Likelihood Estimation	196
	3.4.2. Approximate MLE's	200
	3.4.3. Method of Moments Estimators	204
	3.4.4. Estimation for AR Processes	205
	3.4.5. Other ARMA Estimators	213
3.5.	Identifying ARMA Models	214
	3.5.1. Some Useful Diagnostics	214
	3.5.2. The AIC and Related Criteria	219
	3.5.3. The Stepwise ARMA Method	224

3.5.4.	A Macro for Model Identification	229
3.6.	Box-Jenkins Modeling and Forecasting	230
3.7.	Other Modeling Strategies	240
3.7.1.	Regression with Autoregressive Errors	240
3.7.2.	ARARMA Modeling	243
3.7.3.	State Space and Bayesian Forecasting	246
3.8.	Parametric Spectral Density Estimation	246
3.8.1.	Autoregressive Spectral Estimation	247
3.8.2.	MA and ARMA Spectral Estimation	253
3.9.	Methods for Determining Periodicities	254
3.9.1.	Deterministic Sinusoid Plus Error	255
3.9.2.	Estimating Peak Frequencies in AR Spectra	257
3.10.	Examples and Problems	261

Chapter 4

Analyzing Bivariate Time Series

300

4.1.	Probability Theory for Bivariate Series	301
4.1.1.	Covariance Stationarity	301
4.1.2.	The Spectral Density Function	304
4.1.3.	Bivariate Filters	314
4.1.4.	Filters and Cross-Spectral Analysis	315
4.1.5.	The Bivariate Autoregressive Process	322
4.1.6.	The Bivariate ARMA Process	328
4.1.7.	The Bivariate General Linear Model	329
4.2.	Statistical Inferences for Bivariate Series	329
4.2.1.	Descriptive Statistics	329
4.2.2.	Nonparametric Spectral Density Estimation	334
4.2.3.	Autoregressive Spectral Estimation and Forecasting	339
4.2.4.	Estimating the Transfer Function of a Filter	342
4.3.	Other Topics in Bivariate and Multiple Time Series	345
4.4.	Examples and Problems	345

Appendix A

Some Mathematical and Statistical Topics

357

A.1.	Matrix Operations	358
A.1.1.	The Modified Cholesky Decomposition	358
A.1.2.	The Gram-Schmidt Decomposition	360
A.1.3.	The Sweep Operator	361
A.2.	Fourier Series and Spectral Representations	363
A.2.1.	Fourier Series for Periodic Functions	364
A.2.2.	Spectral Representation of R	367
A.2.3.	Spectral Representation of X	370
A.3.	The Fast Fourier Transform	371

	A.3.1. The FFT Recursion	371
	A.3.2. Reordering Data	372
A.4.	Random Vectors and the Multivariate Normal Distribution	374
	A.4.1. Basic Definitions	374
	A.4.2. Basic Facts	376
	A.4.3. Prediction of a Random Vector	379
	A.4.4. Convergence of Random Variables	386
A.5.	Least Squares Regression Analysis	388
	A.5.1. The General Linear Model	388
	A.5.2. Results for the General Linear Model	390
	A.5.3. The Case of Correlated Errors	395
	A.5.4. Special Types of Regression Analyses	396
	A.5.5. The General Form of the REG Command	405

Appendix B

User's Guide to TIMESLAB 406

B.1.	Hardware and Software Requirements	408
B.2.	Installing and Starting TIMESLAB	409
B.3.	A Guided Tour Through TIMESLAB	412
B.4.	Overview of TIMESLAB Capabilities	421
	B.4.1. Data Types	421
	B.4.2. Command Types	423
	B.4.3. Simple Assignments and Arithmetic Operations	424
	B.4.4. Listing and Deleting Variables	428
	B.4.5. Cases and Types	429
	B.4.6. Error Checking	429
	B.4.7. On-line and User-Definable Help Facilities	430
	B.4.8. Creating and Saving Arrays	431
	B.4.9. Overwriting Previously Defined Variables	433
	B.4.10. Printed Output	433
	B.4.11. The DOS Intraline Editing Functions	433
	B.4.12. The TIMESLAB DOS Mode	434
	B.4.13. The TIMESLAB Full Screen Text Editor	434
	B.4.14. AUTOEXEC.MAC	434
	B.4.15. Random Number Generator Seeds	434
	B.4.16. Keeping a Record of a Session	435
	B.4.17. Exiting from TIMESLAB	435
B.5.	TIMESLAB Graphics	435
	B.5.1. Specifications for Plots	436
	B.5.2. Graphics "Windows"	437
	B.5.3. The Graphics Menu	438
	B.5.4. Graphics Screen Dumps	439
	B.5.5. The FIND Utility	440
	B.5.6. Saving and Retrieving Graphics Screen Images	442

B.5.7.	Scaling and Labeling Graphs	443
B.6.	The TIMESLAB Text Editor	445
B.7.	Using Macros	447
	B.7.1. Basic Features	448
	B.7.2. Macro Branching Commands	452
B.8.	Examples of Writing Macros	453

Appendix C

The TIMESLAB Commands

466

C.1.	Outline of Commands	467
C.2.	Detailed Description of Commands	477

Appendix D

The Data Sets

577

Appendix E

Index of Commands, Theorems, Macros, Examples, and Figures

592

E.1.	Commands	594
E.2.	Theorems	597
E.3.	Macros	599
E.4.	Examples	601
E.5.	Figures	602

References

604

Author Index

614

Subject Index

617

CHAPTER 1

Describing, Transforming, and Forecasting Univariate Time Series

Almost every area of scientific inquiry is concerned with data collected over time, that is, with time series. Figure 1.1 contains the graph of ten such time series. A listing of all of the time series analyzed in this book is included in Appendix D, and the diskettes that accompany the book include a file for each data set.

We have chosen these series to begin our discussion because they illustrate some of the different types of series with which we will be concerned and the types of scientific questions that one can answer using time series analysis. The aim of this chapter is to introduce some of the basic descriptive techniques used in time series analysis and to acquaint the reader with the TIMESLAB software that accompanies the book. We will also begin our discussion of forecasting time series.

1.1. Time Series Data Types

Time series can be classified in many ways, including by the following four characteristics:

1. The dimension of the index set T over which observations are made. The set T can be one-dimensional or multidimensional. Daily births of females in California (Series I) have a one-dimensional index set, while wheat yields recorded over a regular grid of positions in a large field have a two-dimensional index set. This second example also illustrates that the index set need not literally be “time” but can also be “position.”
2. Whether the index set is continuous or discrete. Thus an analog EEG record from one probe is a continuous one-dimensional series, while daily birth data are a discrete one-dimensional series.
3. How many variables are recorded at each element of the index set. Uni-

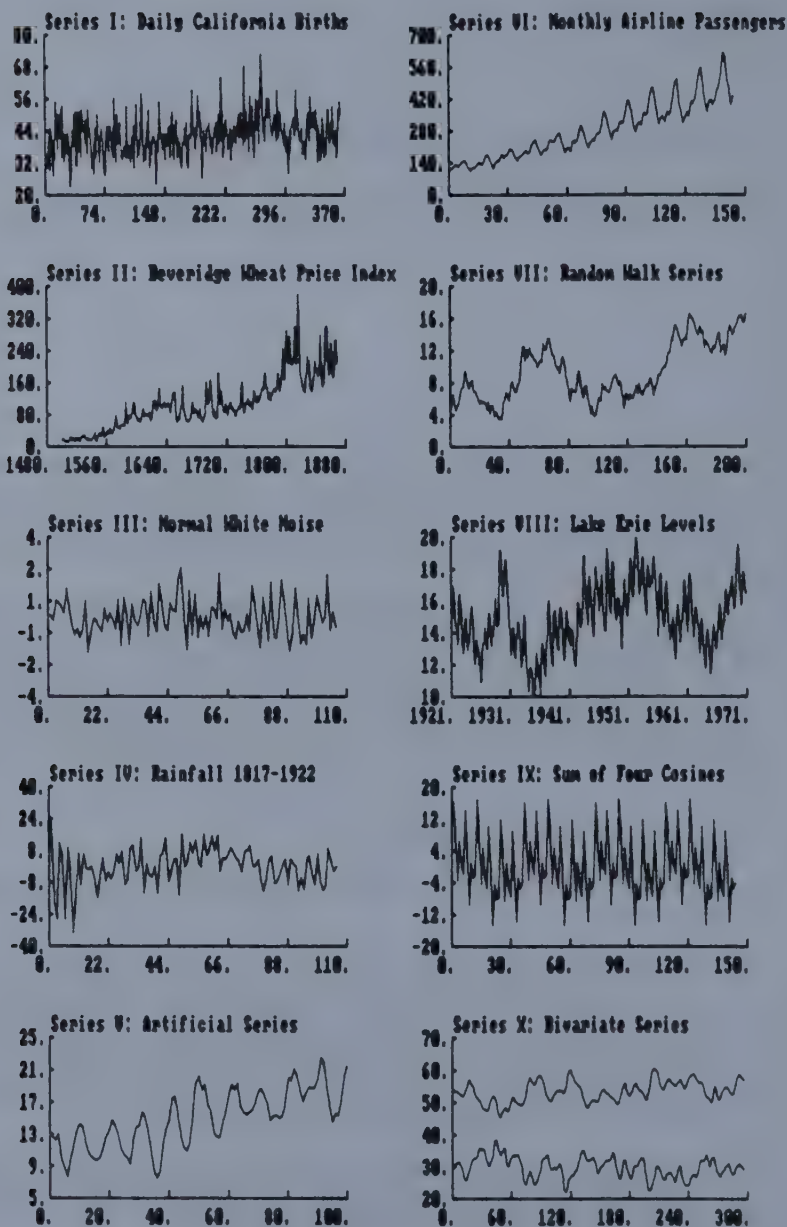


Figure 1.1. Some Examples of Time Series.

variate (also called scalar) time series have one variable at each time, while multivariate time series have a vector of variables measured at each time. For example, a series consisting of monthly interest rates and gross national product is a one-dimensional bivariate time series. Series X is another such example, as the input and output values of a gas in a chemical process are recorded at each time.

4. Whether the variable or variables at each time are themselves discrete or continuous. Most series are continuous; that is, each variable can take on a continuum of values. An example of a discrete valued series is a binary time series (see Example 1.7), one in which an observation can take on only one of two values.

To summarize then, time series can be classified as follows:

1. One or multidimensional (dimension of index set).
2. Discrete or continuous time (nature of index set).
3. Univariate (scalar) or multivariate (number of variables).
4. Discrete or continuous space (nature of variables).

Note that discrete-time time series can also be either equally spaced or irregularly spaced in time. In this book we will be almost exclusively concerned with time series that are one dimensional, equally spaced, discrete in time but continuous in space, and either univariate (Chapters 1–3) or bivariate (Chapter 4).

1.2. Time Series Memory Types

The basic property of time series analysis is that it is concerned with repeated measurements on the same phenomenon at different times or places. Because of this, the analyst must take into account the correlation between successive observations. This is in marked contrast with the data analyzed in elementary statistics courses where one assumes that the data are made up of independent and identically distributed observations obtained by randomly sampling some population or populations. The presence of correlation makes the analysis of time series data and the interpretation of the results much more difficult than in the independent case.

In this section we classify time series into three broad classes according to what we will call their memory type:

1. **Purely Random Series.** This type of series shows no patterns over time. Series III is an example of such a series. It was generated by TIMESLAB

using a random number generator so that it would be indistinguishable from a random sample from a standard normal distribution. Series IV, on the other hand, is a real data set (monthly total rainfall), but appears quite similar to Series III. We will see later that such data are aptly named white noise. Purely random series are also called no-memory series, as one characterization of statistical independence is that an observation at one time has no memory of the observations at any other time.

2. Long-Memory Series. This type of series is the opposite extreme of white noise; that is, a plot of the data looks to be almost that of a deterministic function of time. Series V and VI illustrate this type. The first was artificially generated as values lying on a cosine curve that goes through ten cycles with small random numbers added to each point. Series VI is a real economic time series (monthly total international airline passengers for 12 years). These two series have in common that both could be almost perfectly extrapolated far into the future unless something were to happen to the mechanism generating the data. This is the origin of the term “long memory.” The dependence on the past does not die away quickly. Note that many of the time series in business and economics are long memory.

3. Short-Memory Series. This type lies between white noise and long memory, occurs often in the physical and engineering sciences, and comprises the bulk of the time series that can be most usefully analyzed by the sophisticated methods of time series analysis that we will study. Series VIII and IX appear to be short-memory series; clearly observations close together in time are more similar than those far apart in time, but there are no apparent deterministic patterns in the data (although upon closer inspection you might be able to tell that Series IX is actually the sum of four cosine curves having squared amplitudes 10, 13, 37, and 65 and going through 4, 12, 24, and 48 cycles, see Example 1.4). In a short-memory series the predictability of the observations at one place in time from past observations appears to die out quickly as time goes on.

Time series are usually classified as stationary or nonstationary. We study these concepts in detail in Chapter 2, but for now we will think of nonstationary as long memory and stationary as white noise or short memory.

The WN Command

The TIMESLAB command

```
x=WN(seed,n,dist)
```

will generate a sample realization of length **n** from a white noise series (using the random number generator seed entered in the real scalar argument **seed**)

Table 1.1. Distributions in the WN Command

dist	Distribution	Mean	Variance
1	$N(0, 1) : Z$	0	1
2	$U(0, 1) : U$	1/2	1/12
3	Exponential: $-\log(1 - U)$	1	1
4	Logistic: $\log(U/1 - U)$	0	$\pi^2/3 = 3.28987$
5	Cauchy: $\tan(\pi(U - \frac{1}{2}))$	∞	∞
6	Extreme Value: $\log(-\log(1 - U))$	$-\gamma = -.5772$	$\pi^2/6 = 1.6449$
7	Lognormal: e^Z	$e^{1/2} = 1.6487$	$e^2 - e = 4.67077$
8	Double Exponential	0	2
	$\begin{cases} \log(2U), & U \leq .5 \\ -\log(2(1 - U)), & U > .5 \end{cases}$		

for any of the distributions in Table 1.1.

1.3. Aims of Time Series Analysis

The aims of time series analysis can be roughly grouped into four categories: descriptive, inferential, predictive, and control. Time series analysis is extremely graphical in nature because of the lack of independence among the quantities that it investigates. In the next section we will consider three graphs that can be used for trying to classify a time series into long, short, or no memory.

In Chapter 3 we will consider the inferential part of univariate time series analysis. Essentially we seek to determine from one time series data set what other data could have been observed or are expected to be observed in the future. To do this we must be able to assume that the random mechanism generating the observed data follows some kind of model. These models are introduced in Chapter 2.

The third aim is to use the correlation in time series to predict the future. In Chapter 2 we introduce the basic theory of prediction, and in Chapter 3 we give several examples of prediction methods. Given the ability to predict the future of a time series, one naturally begins to think of controlling that future. Thus one can begin adjusting the values of time series so that the future values of the series are in line with what is desired.

1.4. Basic Descriptive Statistics

The first aim of any statistical procedure is to give a succinct description of the data being analyzed, both graphically and numerically. In time series analysis there are three basic graphical techniques for describing data: the correlogram, the partial correlogram, and the periodogram. In this section we introduce each of these quantities in turn and illustrate how they are used to describe data $x(1), \dots, x(n)$.

1.4.1. The Sample Correlogram

The distinguishing characteristic of a time series is that it can exhibit serial correlation, that is, correlation over time. For example, Figure 1.2 contains scatterplots of $x(t)$ versus $x(t-1)$ for $t = 2, \dots, n$ for Series I and II. Note that for Series I there appears to be little correlation in the plot, while in Series II there appears to be high positive correlation. The macro that was used to produce Figure 1.2 is given in Example 1.3.

In elementary statistics we use the sample correlation coefficient

$$\hat{\rho} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

to measure correlation in a set of bivariate data $(x_1, y_1), \dots, (x_n, y_n)$. In a time series $x(1), \dots, x(n)$, we usually want to measure the correlation of the data with themselves except “lagged” a certain number of time units. Thus for a lag v , we have $n - v$ pairs of x ’s that are separated by v time units, namely the pairs

$$(x(1), x(1+v)), (x(2), x(2+v)), \dots, (x(n-v), x(n)).$$

One way to measure correlation in these pairs would be to apply the above formula for the sample correlation coefficient. However, the traditional way to measure serial correlation in time series is by the sample autocorrelation coefficient as given in the following definition. In Section 3.1.2 we discuss why this definition is used.

Definition. Let $x(1), \dots, x(n)$ be a univariate time series data set. The sample autocorrelation coefficient of lag v is given by

$$\hat{\rho}(v) = \frac{\sum_{t=1}^{n-v} (x(t) - \bar{x})(x(t+v) - \bar{x})}{\sum_{t=1}^n (x(t) - \bar{x})^2}, \quad v < n,$$

where \bar{x} is the sample mean of $x(1), \dots, x(n)$. A plot of $\hat{\rho}(v)$ versus v for $v = 0, 1, \dots, M$ for some maximum lag M is called the correlogram of the data.

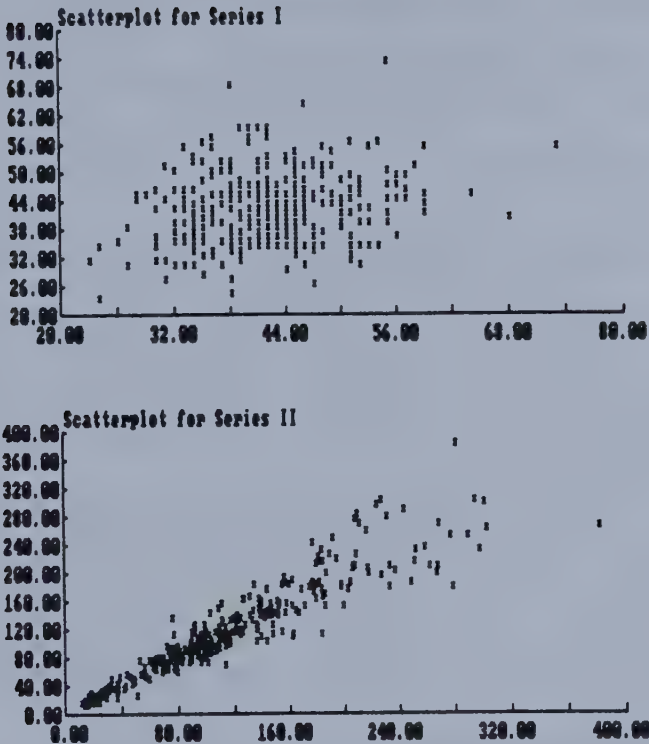


Figure 1.2. Scatterplot of $x(t)$ versus $x(t-1)$ for Series I and II.

Note that $\hat{\rho}(0) = 1$ and that the largest value of v that can be considered is $n-1$, since if $v \geq n$, there are no pairs of x 's separated by v time units. Note further that we will define $\hat{\rho}(-v) = \hat{\rho}(v)$ and thus will sometimes write

$$\hat{\rho}(v) = \frac{\sum_{t=1}^{n-|v|} (x(t) - \bar{x})(x(t+|v|) - \bar{x})}{\sum_{t=1}^n (x(t) - \bar{x})^2}, \quad |v| < n.$$

Suppose that we are considering a bivariate data set; that is, at each time we observe

$$\mathbf{z}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \quad t = 1, \dots, n.$$

Then we have the following definition.

Definition. The cross-correlation coefficient of lag v is given by

$$\hat{\rho}_{xy}(v) = \begin{cases} \frac{\sum_{t=1}^{n-v} (x(t) - \bar{x})(y(t+v) - \bar{y})}{\sqrt{\sum_{t=1}^n (x(t) - \bar{x})^2 \sum_{t=1}^n (y(t) - \bar{y})^2}}, & v = 0, \dots, n-1 \\ \hat{\rho}_{yx}(-v), & v = -(n-1), \dots, 0. \end{cases}$$

Note that $\hat{\rho}_{xy}$ is not symmetric about $v = 0$. A plot of $\hat{\rho}_{xy}(v)$ versus v for $v = -M, \dots, M$ is called the cross-correlogram of x and y .

Correlograms are used for three basic purposes. To illustrate these purposes, consider Figure 1.3 where we have given the correlograms for the ten series introduced earlier. A large value of $\hat{\rho}(v)$ is indicative of a possible periodicity in the data of v time units. For example, the correlogram of a pure cosine curve is also sinusoidal (see Problem T1.6). Notice that Series V and VI exhibit this behavior.

A correlogram that does (does not) decay rapidly to zero and then stay there as v increases means that the series is short (long) memory. Note that the wheat price index, the artificial series, the airline data, and the random walk series appear to be long memory by inspection of the correlogram.

A large cross-correlation of lag v indicates that x and y may be very similar except that they are out of phase; that is, one of the series is leading the other. This would appear to be the case in the bivariate series that we have been considering.

The CORR and CORR2 Commands

The TIMESLAB command CORR can be used to calculate the correlogram of a univariate time series, while CORR2 finds auto- and cross-correlations for a bivariate series. Thus the following sequences of commands will produce a plot of the correlogram of a series x and the cross-correlogram for series x and y :

;Correlogram of x:

```
rho=CORR(x,n,M,0,1,R0,per)
PLOT(rho,M,0,M,-1,1)
```

;Cross correlogram of x and y:

```
CORR2(x,y,n,M,1,Rx0,Ry0,rhoxy0,rhox,rhoxy,rhoxy)
rr=REVERSE(rhoxy,M)
rho=<rr,rhoxy0,rhoxy>
MM=-M
ind=LINE(M,0,1)
ind1=-ind
ind1=REVERSE(ind1,M)
```

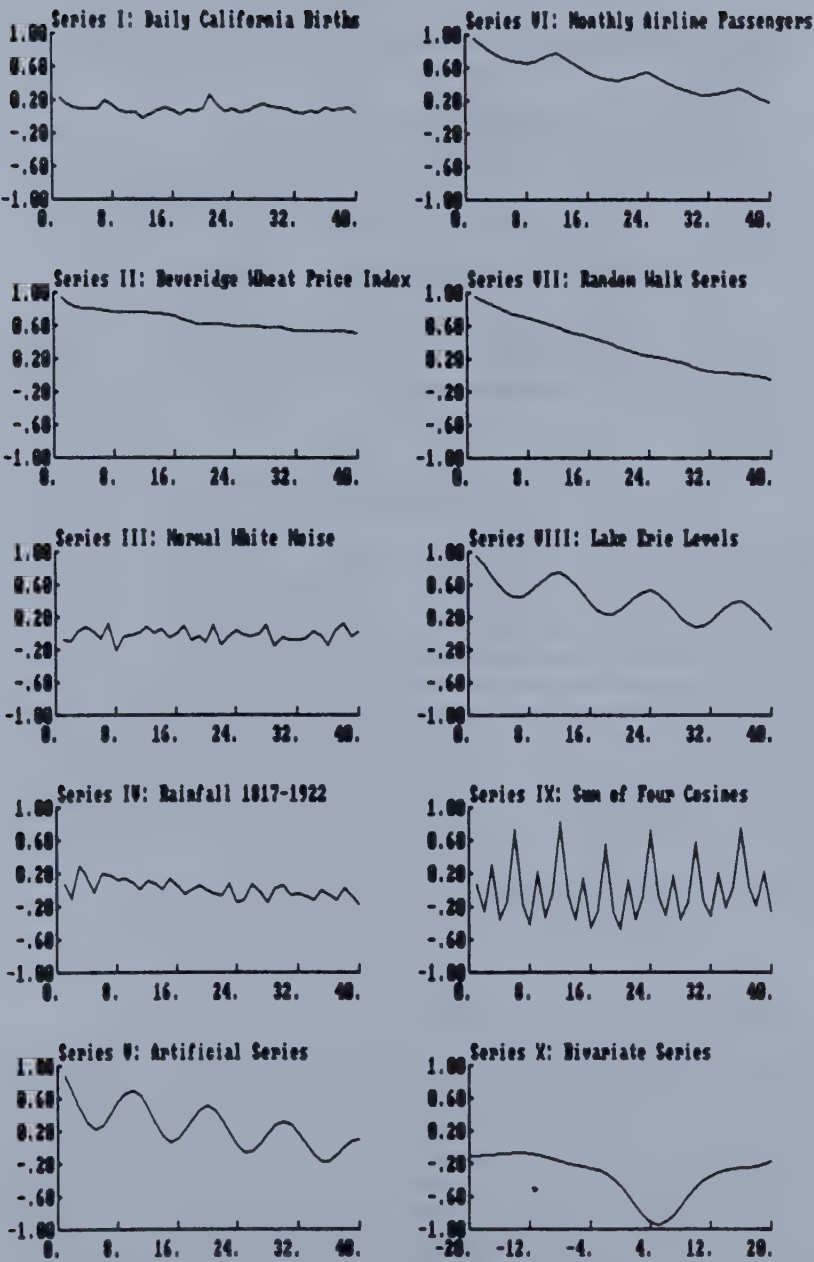


Figure 1.3. The Correlograms for Series I-X.

```

ind=<ind1,0,ind>
m2p1=2*M
m2p1=m2p1+1
LABEL(rho)='Cross-Correlogram'
PLOT(ind,rho,m2p1,mm,M,-1,1)

```

We will discuss CORR and CORR2 in further detail in this chapter.

1.4.2. The Sample Partial Correlogram

As in elementary statistics, high correlation does not imply causality. We can introduce partial and multiple correlations in an attempt to describe further the correlation in certain sets of variables. In elementary statistics, if we have random samples from populations Y , Z , and X_1, \dots, X_p , we can find the residuals e_Y and e_Z of regressing Y on the X 's and Z on the X 's and then find the usual sample correlation coefficient between $e_Y(1), \dots, e_Y(n)$ and $e_Z(1), \dots, e_Z(n)$. The result is what is called the sample partial correlation coefficient between Y and Z given X_1, \dots, X_p , and is often denoted $\hat{\rho}_{YZ|X_1 \dots X_p}$. The sample multiple correlation coefficient $R^2_{Y|X_1 \dots X_p}$ is the proportion of variability in Y that is 'explained' by its linear relationship with the X 's (see Section A.5 for more detail about the multiple correlation coefficient). The analogous quantities in univariate time series analysis are the sample partial autocorrelation coefficients and residual variances. The sample partial autocorrelation coefficient of lag v is the correlation between $x(t)$ and $x(t+v)$ after having removed the common linear effect of the data in between (the lag one partial is just the usual lag one sample autocorrelation). We denote the lag v partial autocorrelation by $\hat{\theta}_v$. In Section 3.4.4 we discuss in more detail the exact definition of $\hat{\theta}_v$.

The PARCORR Command

The command

```
part=PARCORR(x,n,M[,rvar])
```

will calculate the first M sample partial autocorrelations (returned in the array **part**) and optionally the corresponding sample residual variances $\hat{\sigma}_1^2, \dots, \hat{\sigma}_m^2$ in the array **rvar** where $\hat{\sigma}_v^2$ is the sample variance of the residuals of having regressed $x(t)$ on the previous v x 's. Thus if we divide **rvar** by the sample variance of the x 's and then subtract the result from one, we obtain the time series analog of a multiple correlation coefficient. We will usually display what we call the standardized residual variances, that is, the residual variances divided by the sample variance. These values will always be between zero and one, and a small (large) value indicates that $x(t)$ is (is not) very predictable from its past, thus indicating that the data are long (short) memory. A useful rule

of thumb is that data are long memory if their standardized residual variance sequence becomes less than $8/n$ for some lags.

In Figure 1.4 we display the partial correlogram and standardized residual variances for each of the univariate series that we have been considering. Since the two graphs for each series are on scales of $(-1,1)$ and $(0,1)$, respectively, we can put them both on the same axes using the `PLOT2` command with the partials indicated by a solid curve and the standardized residual variances indicated by the broken curve. We can use the partial autocorrelations in the same way that usual partial correlations are used. In Series VII, for example, the correlogram takes some time to decay to zero, while the partial correlogram is large for lag one and then small henceforth. This indicates that the correlation between $x(t)$ and $x(t+2)$ is due only to the common relationship of $x(t)$ and $x(t+2)$ to $x(t+1)$.

1.4.3. The Periodogram and Sample Spectral Density

A recurring theme in science is the study of deterministic phenomena that have been observed over time (music, speech, radio signals, etc.) via harmonic analysis, that is, by decomposing a time record into the sum of sinusoids of various frequencies and amplitudes. Harmonic analysis is also important in the study of random phenomena observed over time. We will study the theoretical aspects of this analysis in Chapter 2 (see also Section A.2), but in this section we will motivate the theory by looking at what is called the periodogram of a data set $x(1), \dots, x(n)$.

The basic idea of harmonic analysis is that one can find a unique set of sinusoids (a cosine plus sine of the same frequency) that when added together reconstruct the mathematical object being studied. Thus one can study a complicated object by breaking it up into these simple “frequency components” and studying them separately. Further, in a certain sense, conclusions made about one sinusoid are independent of conclusions made about others.

A similar idea in statistics is that of orthogonal contrasts in the analysis of variance, wherein the results of applying different treatments to experimental units can be decomposed into the sum of linear, quadratic, and higher-order effects that are orthogonal, or statistically independent.

The mathematical objects that can be studied by harmonic analysis vary widely. Examples include sequences of finitely many numbers, infinitely long sequences, functions that are periodic, and general continuous or even discontinuous functions. No matter what object is being studied, the results are always of the form of a decomposition into independent sinusoids that can be studied separately.

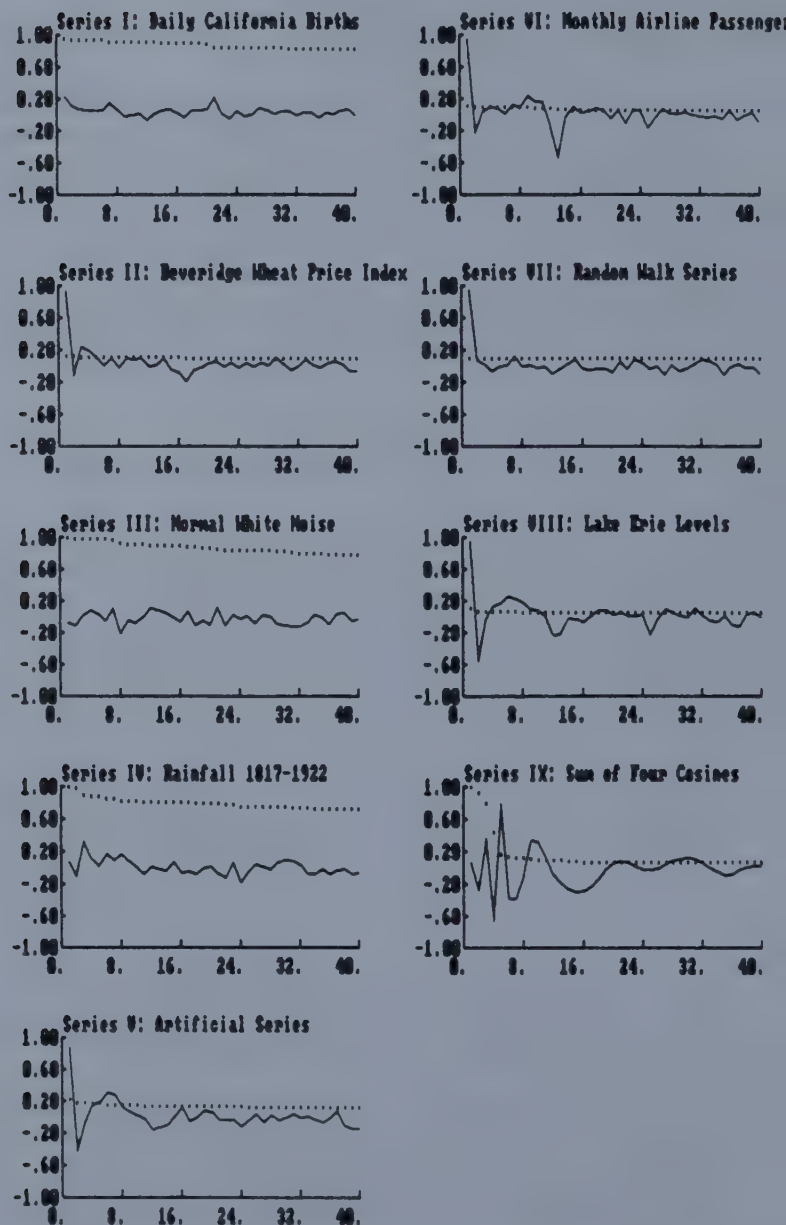


Figure 1.4. The Partial Autocorrelations (Solid Curve) and Standardized Residual Variances (Dotted Curve) for Series I-IX.

Definition. A sinusoid $g_k(x)$ of period k (or frequency $1/k$) is the function

$$g_k(x) = a \cos \frac{2\pi x}{k} + b \sin \frac{2\pi x}{k}, \quad x \in (-\infty, \infty).$$

Since \cos and \sin have period 2π , we have that $g_k(x + kl) = g_k(x)$, for any integer l . We can also write (see Problem T1.2)

$$g_k(x) = C \cos\left(\frac{2\pi x}{k} - \phi\right),$$

where $C = \sqrt{a^2 + b^2}$ and $\phi = \arctan(b/a)$ are called the amplitude and phase of g_k .

The COS and SIN Commands

The TIMESLAB commands $\mathbf{x} = \text{COS}(\mathbf{n}, \mathbf{a}, \mathbf{p})$ and $\mathbf{y} = \text{SIN}(\mathbf{n}, \mathbf{a}, \mathbf{p})$ generate arrays \mathbf{x} and \mathbf{y} where

$$\mathbf{x}(j) = a \cos \frac{2\pi(j-1)}{p}, \quad j = 1, \dots, n,$$

$$\mathbf{y}(j) = a \sin \frac{2\pi(j-1)}{p}, \quad j = 1, \dots, n,$$

and \mathbf{a} and \mathbf{p} are real scalars with $\mathbf{p} \neq 0$.

A data set $x(1), \dots, x(n)$ is an example of a finite sequence of numbers. The basic fact of the harmonic analysis of such a sequence is that we can calculate from $x(1), \dots, x(n)$ a set of orthogonal sinusoids of frequencies $1/n, 2/n, \dots, [n/2]/n$, where $[c]$ denotes the greatest integer less than or equal to c , together with a constant, so that when the constant and the sinusoids are added together, the resulting function coincides with the data at the indices $1, 2, \dots, n$. Further, the sample variance of the x 's is proportional to the sum of the squared amplitudes of the sinusoids.

We will formalize these results in Theorem 1.4.2, but first we have the following theorem, which describes the orthogonality of sinusoids.

Theorem 1.4.1	ORTHOGONALITY OF SINUSOIDS
----------------------	-----------------------------------

Let $\omega_k = (k-1)/n$, for $k = 1, \dots, [n/2] + 1$. Then

$$a) \sum_{t=1}^n \cos 2\pi(t-1)\omega_j \cos 2\pi(t-1)\omega_k = \begin{cases} 0, & \omega_j \neq \omega_k \\ n, & \omega_j = \omega_k = 0, .5 \\ n/2, & \omega_j = \omega_k \neq 0, .5 \end{cases}$$

$$b) \sum_{t=1}^n \sin 2\pi(t-1)\omega_j \sin 2\pi(t-1)\omega_k = \begin{cases} n/2, & \omega_j = \omega_k \neq 0, .5 \\ 0, & \text{otherwise} \end{cases}$$

$$c) \sum_{t=1}^n \cos 2\pi(t-1)\omega_j \sin 2\pi(t-1)\omega_k = 0, \quad \text{all } \omega_j, \omega_k$$

$$d) \sum_{t=1}^n \cos 2\pi(t-1)\omega_j = \begin{cases} n, & \omega_j = 0, .5 \\ 0, & \text{otherwise} \end{cases}$$

$$e) \sum_{t=1}^n \sin 2\pi(t-1)\omega_j = 0, \quad \text{all } \omega_j$$

Proof: We include the proof because it illustrates techniques used in many theorems in harmonic analysis. Letting $\omega_k = 0$ in part (a) gives part (d), and letting $\omega_j = 0$ in part (c) gives part (e), since $\cos(0) = 1$. We illustrate the other parts by proving part (a). Recall the identities

$$e^{i\theta} = \cos \theta + i \sin \theta, \quad \cos \alpha \cos \beta = \frac{\cos(\alpha + \beta) + \cos(\alpha - \beta)}{2},$$

and denote the real and imaginary parts of the complex number $z = a + bi$ by $a = \operatorname{Re}(z)$ and $b = \operatorname{Im}(z)$. Then

$$\begin{aligned} S_{j,k} &= \sum_{t=1}^n \cos 2\pi(t-1)\omega_j \cos 2\pi(t-1)\omega_k \\ &= \frac{1}{2} \left[\sum_{t=1}^n \cos 2\pi(t-1)(\omega_j + \omega_k) + \sum_{t=1}^n \cos 2\pi(t-1)(\omega_j - \omega_k) \right] \\ &= \frac{1}{2} \operatorname{Re} \left[\sum_{t=1}^n e^{2\pi i(t-1)(\omega_j + \omega_k)} + \sum_{t=1}^n e^{2\pi i(t-1)(\omega_j - \omega_k)} \right] \\ &= \frac{1}{2} \operatorname{Re} \left[\sum_{l=0}^{n-1} r_1^l + \sum_{l=0}^{n-1} r_2^l \right], \end{aligned}$$

where $r_1 = e^{2\pi i(\omega_j + \omega_k)}$ and $r_2 = e^{2\pi i(\omega_j - \omega_k)}$. For a general geometric series with base r , we have

$$\sum_{l=0}^{n-1} r^l = \begin{cases} \frac{1 - r^n}{1 - r}, & r \neq 1 \\ n, & r = 1. \end{cases}$$

Note that r_1 and r_2 are both of the form $e^{2\pi i\theta} = \cos 2\pi\theta + i \sin 2\pi\theta$, which equals one only if θ is an integer. But since $\omega_j \in [0, .5]$, the only way $\omega_j + \omega_k$

can be an integer is if $\omega_j = \omega_k = 0$ or .5, while $\omega_j - \omega_k$ can be an integer only if $\omega_j = \omega_k$. Thus we have

$$S_1 = \sum_{l=0}^{n-1} r_1^l = \begin{cases} n, & \omega_j = \omega_k = 0, .5 \\ \frac{1 - r_1^n}{1 - r_1}, & \omega_j \neq \omega_k \text{ or if } \omega_j = \omega_k \neq 0, .5 \end{cases}$$

$$S_2 = \sum_{l=0}^{n-1} r_2^l = \begin{cases} n, & \omega_j = \omega_k \\ \frac{1 - r_2^n}{1 - r_2}, & \omega_j \neq \omega_k. \end{cases}$$

But $r_1^n = e^{2\pi i(\omega_j + \omega_k)n} = e^{2\pi i(j+k-2)} = 1$, and r_2^n is also one, so that the second terms for S_1 and S_2 are both zero. Thus,

$$S_{j,k} = \frac{1}{2} \text{Re}(S_1 + S_2)$$

$$= \begin{cases} n, & \omega_j = \omega_k = 0, .5 \\ \frac{n}{2}, & \omega_j = \omega_k \neq 0, .5 \\ 0, & \omega_j \neq \omega_k, \end{cases}$$

which was to be proved.

Before presenting the sinusoidal decomposition of a time series data set, we define the discrete Fourier transform of a set of numbers.

Definition. The discrete Fourier transform (DFT) of the (possibly complex) numbers $x(1), \dots, x(n)$ is the set of complex numbers $z(1), \dots, z(n)$ given by

$$z(k) = \sum_{t=1}^n x(t) e^{2\pi i(t-1)\omega_k}$$

$$= \sum_{t=1}^n x(t) \cos 2\pi(t-1)\omega_k + i \sum_{t=1}^n x(t) \sin 2\pi(t-1)\omega_k,$$

where $\omega_k = (k-1)/n$, $k = 1, \dots, n$.

If we insert a minus sign into the exponent in the first expression, we have what is called the inverse discrete Fourier transform (IDFT), since if one were to find the IDFT of the DFT of x , the result would be nx .

The basic facts about the harmonic analysis of $x(1), \dots, x(n)$ are given in the next theorem, the results of which can be shown using Theorem 1.4.1 and further trigonometric identities.

Theorem 1.4.2 SINUSOIDAL DECOMPOSITION OF DATA

Let $x(1), \dots, x(n)$ be a finite sequence of numbers and let $z(1), \dots, z(n)$ be the DFT of x . For $k = 1, \dots, n$, let $\omega_k = (k-1)/n$ and define

$$a_k = \frac{1}{n} \operatorname{Re}(z(k)), \quad b_k = \frac{1}{n} \operatorname{Im}(z(k)).$$

Let $\hat{\sigma}^2 = \frac{1}{n} \sum_{t=1}^n (x(t) - \bar{x})^2$, and for $t = 1, \dots, n$ define

$$\begin{aligned} g_k(t) &= a_k \cos 2\pi(t-1)\omega_k + b_k \sin 2\pi(t-1)\omega_k \\ &= C_k \cos[2\pi(t-1)\omega_k - \arctan(b_k/a_k)]. \end{aligned}$$

Then

$$a) \ a_1 = \bar{x}, \quad b_1 = 0, \quad C_1^2 = \bar{x}^2, \quad g_1(t) = \bar{x}$$

$$b) \ x(t) - \bar{x} = \sum_{k=2}^n g_k(t)$$

$$c) \ \hat{\sigma}^2 = \sum_{k=2}^n C_k^2$$

d) For $k = 2, \dots, [n/2] + 1$, we have

$$a_k = a_{n-k+2}, \quad b_k = -b_{n-k+2}$$

$$\cos 2\pi(t-1)\omega_k = \cos 2\pi\omega_{n-k+2}$$

$$\sin 2\pi(t-1)\omega_k = -\sin 2\pi\omega_{n-k+2}$$

and thus

$$g_k(t) = g_{n-k+2}(t), \quad C_k^2 = C_{n-k+2}^2,$$

which gives

$$x(t) - \bar{x} = \begin{cases} 2 \sum_{k=2}^{[n/2]+1} g_k(t), & n \text{ odd} \\ 2 \sum_{k=2}^{n/2} g_k(t) + g_{(n/2)+1}(t), & n \text{ even} \end{cases}$$

$$\hat{\sigma}^2 = \begin{cases} 2 \sum_{k=2}^{[n/2]+1} C_k^2, & n \text{ odd} \\ 2 \sum_{k=2}^{n/2} C_k^2 + C_{(n/2)+1}^2, & n \text{ even.} \end{cases}$$

e) For $k = 1, \dots, [n/2] + 1$, the vectors $\mathbf{g}_k = (g_k(1), \dots, g_k(n))^T$ are orthogonal, that is, $\mathbf{g}_j^T \mathbf{g}_k = 0$, $j \neq k$.

Implications: This theorem gives a decomposition of a time series data set into a set of orthogonal (part (e)) sinusoids of frequencies $1/n, 2/n, \dots, [n/2]/n$ (part (d)), and gives a decomposition of the variability in the x 's (as measured by $\hat{\sigma}^2$) in terms of the sum of squared amplitudes of the sinusoids (part (c)). In terms of understanding why the x 's vary then, these squared amplitudes play an important role. Note that $C_k^2 = |z(k)|^2/n^2$.

Definition. For a time series data set $x(1), \dots, x(n)$, let $\omega_k = (k-1)/n$, $k = 1, \dots, [n/2] + 1$, and define

$$C_k^2 = \frac{1}{n^2} \left| \sum_{t=1}^n x(t) e^{2\pi i(t-1)\omega_k} \right|^2, \quad k = 1, \dots, [n/2] + 1.$$

A plot of nC_k^2 versus ω_k is called the periodogram of x . The function

$$\hat{f}(\omega) = \begin{cases} \frac{1}{n} \left| \sum_{t=1}^n x(t) e^{2\pi i(t-1)\omega} \right|^2, & \omega \in [0, .5] \\ \hat{f}(1-\omega), & \omega \in [.5, 1] \end{cases}$$

is called the sample spectral density function of x .

Note that the periodogram is the sample spectral density evaluated at the so-called natural frequencies $\omega_1, \omega_2, \dots, \omega_{[n/2]+1}$.

Interpreting the Periodogram

In Figure 1.5 we give a plot of three data sets of length 200 (and the log of their periodograms) that were constructed by

$$x(t) = \alpha \cos \frac{2\pi(t-1)}{100} + \beta \cos \frac{2\pi(t-1)}{10} + \delta \cos \frac{2\pi(t-1)}{4},$$

that is, as the sum of pure cosines of frequencies $1/100$, $1/10$, and $1/4$. The three series were obtained by varying α , β , and δ (Series 1 has (10,3,1), Series 2 has (3,3,3), and Series 3 has (1,3,10)).

Recall that a sinusoid of frequency ω is the sum of a sine and a cosine of that frequency, but for simplicity we are considering sinusoids that have no sine part. A sinusoid of long period (low frequency) is very smooth in appearance relative to one of short period (high frequency). Thus when α is large relative

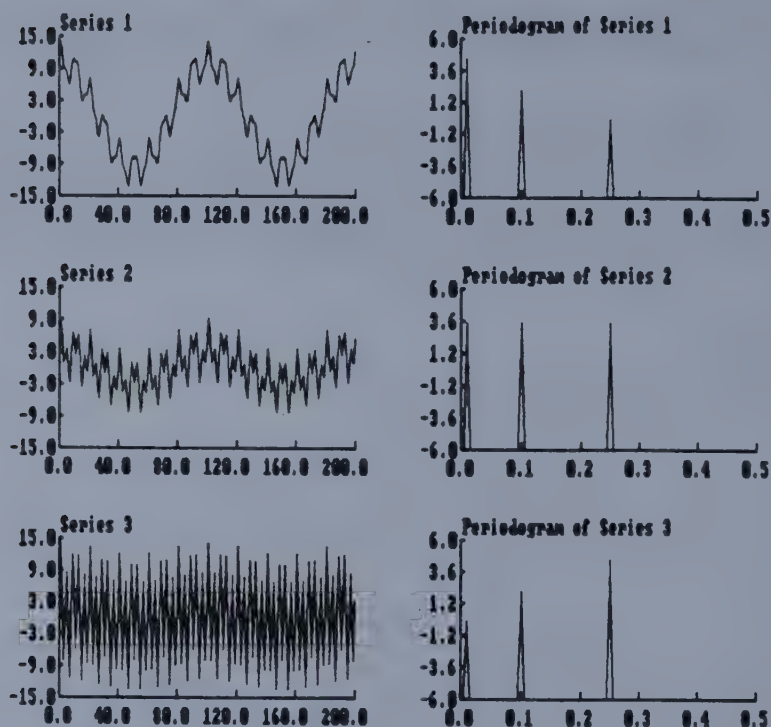


Figure 1.5. Sums of Pure Cosines and Their log Periodograms.

to β and γ (such as Series 1), we would expect x to be relatively smooth in appearance; that is, the long-term rise and fall of the data should be large relative to short term oscillations. On the other hand, Series 3 appears quite wiggly since γ is large relative to α and β . In Series 2 we have used equal values of the three coefficients and the resulting data are between Series 1 and 3 in terms of wiggleness.

Note how the periodograms pick out the amplitudes of the component sinusoids. In general, of course, time series data sets are not made up of sums of only a few pure sinusoids. However, with the above example as motivation, we can make the qualitative statements given in Table 1.2.

In Figure 1.6 we give plots of the logarithm of a standardized form of the periodograms of the nine univariate series that we have been discussing throughout this chapter. Note how these graphs confirm the discussion above. In particular, note the harmonics in the periodogram of the airline data, the random pattern in those for Series III and IV, the excess of low frequency in

Table 1.2. Interpreting the Periodogram

Appearance of Data	Nature of Periodogram
Smooth	Excess of low frequency; that is, amplitudes of sinusoids of low frequency (long period) are large relative to other frequencies
Wiggly	Excess of high frequency
Random (no pattern)	No frequencies dominate
Basically sinusoidal of period p time units	A peak at frequency $1/p$
Periodic of period p but not sinusoidal	A peak at fundamental frequency $1/p$ and peaks at some multiples of $1/p$ (harmonics) (see Problem C1.6 for an example of this)

Series VII, the periodicity in the birth data, and how the periodogram is able to tell that Series IX is in fact the sum of four pure cosines.

Displaying the Periodogram

It often happens that a few values of a periodogram are very large relative to the rest and thus dwarf them in the plot. Plotting the (natural) log of the values rather than the values themselves allows us to see other possible values of interest. In Chapter 3 we will see other reasons for plotting the log (see Theorem 3.1.4). We would also like to plot the log periodogram on some standard scale so that several such plots can be compared in a meaningful way. A useful standardization of the periodogram is suggested by the following theorem, which follows immediately from part (c) of Theorem 1.4.2.

Theorem 1.4.3

STANDARDIZING THE PERIODOGRAM

Let nC_k^2 for $k = 1, \dots, n$ be the periodogram ordinates of a time series $x(1), \dots, x(n)$, and let $\hat{\sigma}^2 = \frac{1}{n} \sum_{t=1}^n (x(t) - \bar{x})^2$. Then

$$\frac{1}{n} \sum_{k=2}^n \frac{nC_k^2}{\hat{\sigma}^2} = 1;$$

that is, the average value of $nC_k^2/\hat{\sigma}^2$ is one.

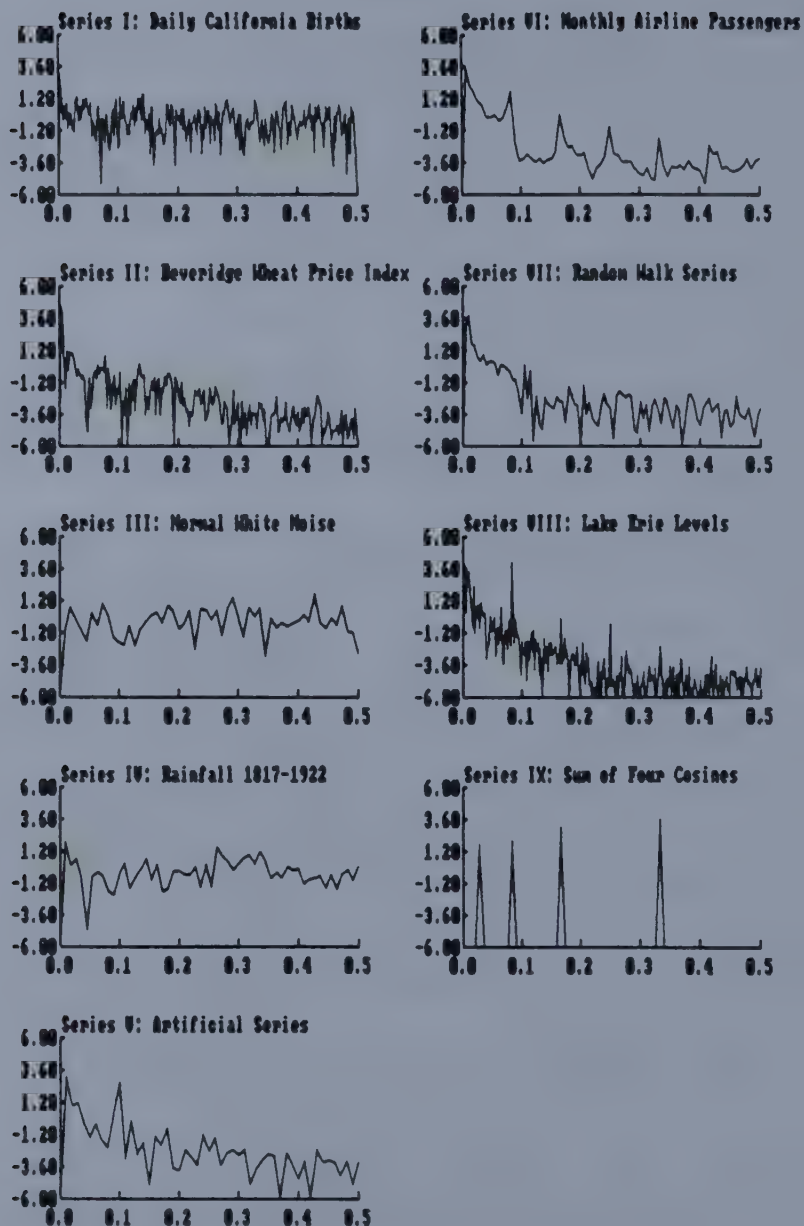


Figure 1.6. The Log of the Standardized Periodograms of Series I-IX.

Because of this theorem we will routinely display

$$\log \left(\frac{nC_k^2}{\hat{\sigma}^2} \right) \text{ versus } \frac{k-1}{n}, \quad k = 1, \dots, [n/2] + 1,$$

with the graph truncated at -6 and 6 on the vertical axis. We will rarely encounter a value of $nC_k^2/\hat{\sigma}^2$ that is greater than e^6 or less than e^{-6} .

The PLOTSP and PLOTCSF Commands

If one has an array \mathbf{f} of length $[Q/2]+1$ and a real scalar c , then the command

PLOTSP(\mathbf{f}, Q, c)

will produce a plot of $\log(\mathbf{f}(j)/c)$ versus $(j-1)/Q$ for $j = 1, \dots, [Q/2] + 1$, so that the vertical scale runs from -6 to 6 . The command

PLOTCSF(\mathbf{f}, Q)

will produce a plot of $\mathbf{f}(j)$ versus the same set of frequencies with no truncation. Thus one can use **PLOTSP** (with c being $\hat{\sigma}^2$) to get the standard periodogram plot that we have been discussing, or use **PLOTCSF** to get a plot of the actual periodogram. These two commands will be used extensively in the sequel, as we will be introducing other functions that will be similar to the periodogram.

It is important to note that most of the **TIMESLAB** commands that produce arrays to be plotted via **PLOTSP** or **PLOTCSF** will have Q (the number of frequencies between 0 and 1) as an argument but will produce only $[Q/2]+1$ values because the quantities that are calculated are symmetric about frequency .5.

The Sample Spectral Distribution Function

We have seen that data that have no obvious trends, cycles, or serial correlation should have a periodogram that oscillates roughly about a constant. On the other hand, smooth (wiggly) data have an excess of low (high) frequency, while cyclic data will have peaks in their periodogram. One useful way to display these ideas graphically is via the sample spectral distribution function.

Definition. Let $\hat{f}(\omega_1), \dots, \hat{f}(\omega_Q)$ be the sample spectral density function of data $x(1), \dots, x(n)$ at the frequencies $\omega_j = (j-1)/Q$, $j = 1, \dots, Q$. Let $q = [Q/2] + 1$ and

$$\hat{F}(\omega_k) = \frac{\sum_{j=1}^k \hat{f}(\omega_j)}{\sum_{j=1}^q \hat{f}(\omega_j)}, \quad k = 1, \dots, q.$$

Then $\hat{F}(\omega_1), \dots, \hat{F}(\omega_q)$ is called the sample spectral distribution function of x .

If $Q = n$, then the \hat{f} 's are in fact the periodogram ordinates, and thus the \hat{F} 's could be called the cumulative periodogram of x . Note that $\hat{F}(\omega_q) = 1$, while if $\bar{x} = 0$, we have $\hat{f}(\omega_1) = 0$ and so $\hat{F}(\omega_1) = 0$. Thus

$$0 \leq \hat{F}(\omega_k) \leq 1, \quad k = 1, \dots, q,$$

and the spectral distribution function of random data will fluctuate about the straight line that connects the points (0,0) and (.5,1), that is, the line $y = 2x$.

The CUMSP Command

The command

F=CUMSP(f,Q)

will compute the sample spectral distribution function F corresponding to the sample spectral density function f . Again, note that Q is the number of frequencies between 0 and 1 at which f has been calculated, while the arrays F and f are of length $q = [Q/2] + 1$. Once F has been found from CUMSP, then PLOTCSF can be used to plot it.

In Figure 1.7 we display the cumulative spectral density of our example data sets. Note that these graphs are much less variable than their corresponding log periodograms. Also, an excess of low (high) frequency means that the cumulative periodogram starts out above (below) the line $y = 2x$ before catching up to it before $\omega = .5$, while peaks in the periodogram translate into jumps in the cumulative periodogram (such as at frequency 1/12 in the airline data), and the cumulative periodograms of the white noise series (Series III and IV) vary little from the line $y = 2x$.

1.4.4. The Relationship of the Correlogram and Periodogram

Recall that the sample autocorrelation coefficient of lag v for data x is given by

$$\hat{\rho}(v) = \frac{\sum_{t=1}^{n-|v|} (x(t) - \bar{x})(x(t+v) - \bar{x})}{\sum_{t=1}^n (x(t) - \bar{x})^2}, \quad |v| < n.$$

Then $\hat{\rho}(v)$ can be written as

$$\hat{\rho}(v) = \frac{\hat{R}(v)}{\hat{R}(0)}, \quad |v| < n,$$

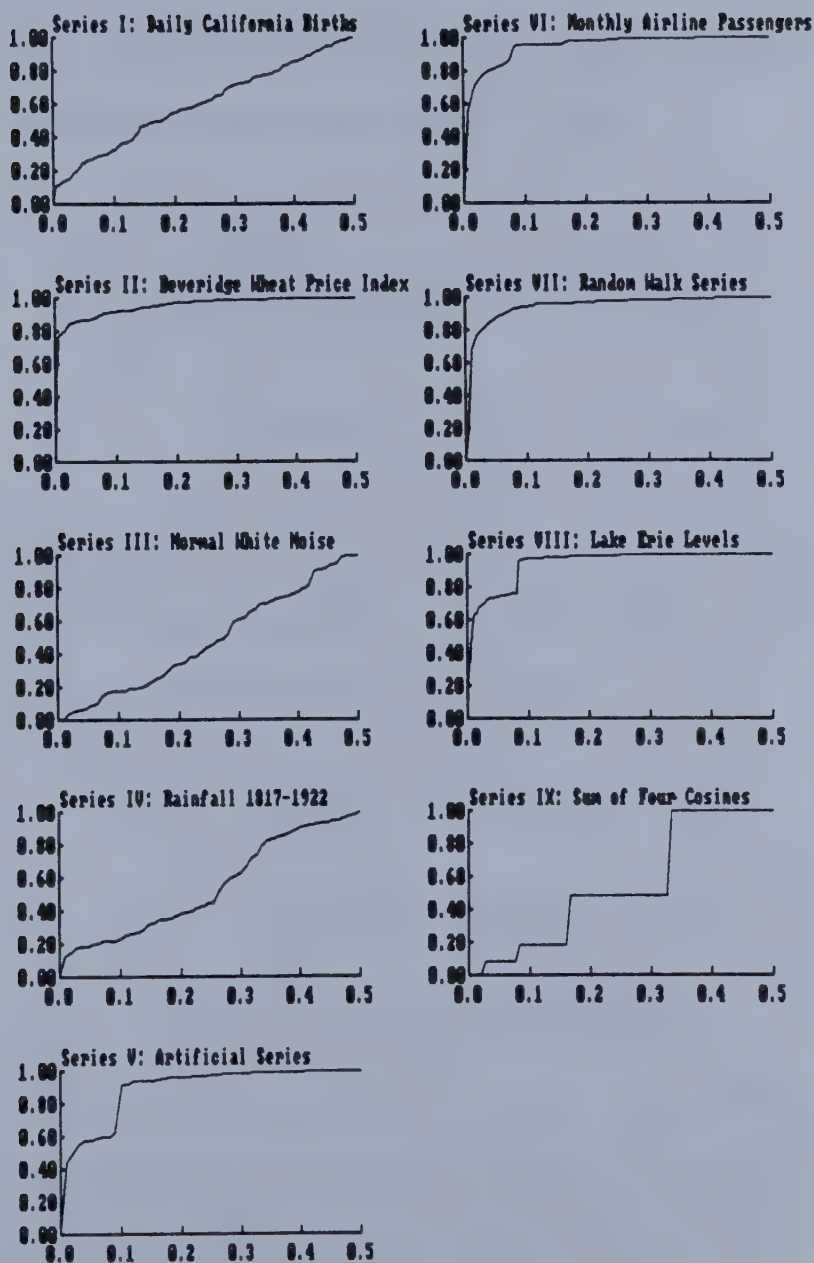


Figure 1.7. Cumulative Periodograms for Series I-IX.

if we define the sample autocovariance function \hat{R} by

$$\hat{R}(v) = \frac{1}{n} \sum_{t=1}^{n-|v|} (x(t) - \bar{x})(x(t+v) - \bar{x}), \quad |v| < n.$$

We show in this section various relationships among $\hat{\rho}$, \hat{R} , and the sample spectral density \hat{f} . We also describe how an algorithm called the fast Fourier transform (FFT) can be used to calculate the correlogram and periodogram much more rapidly than a straightforward implementation of their defining formulas. These results are summarized in the next theorem.

Theorem 1.4.4 RELATIONS AMONG $\hat{\rho}$, \hat{R} , \hat{f}

Let $x(1), \dots, x(n)$ be a time series data set, and for $t = 1, \dots, n$, let $z(t) = x(t) - \bar{x}$. For a positive integer M , let

$$y(t) = \begin{cases} z(t), & t = 1, \dots, n \\ 0, & t = n+1, \dots, n+M; \end{cases}$$

that is, the series y is just the series z “padded” with M zeros. Let $\hat{\rho}_z$, \hat{R}_z , and \hat{f}_z be the autocorrelation, autocovariance, and sample spectral density of z , respectively, and let \hat{f}_y be the sample spectral density of y . Then

$$a) \hat{f}_z(\omega) = \sum_{|v| < n} \hat{R}_z(v) e^{-2\pi i v \omega}, \quad \omega \in [0, 1]$$

$$b) \hat{R}_z(v) = \int_0^1 \hat{f}_z(\omega) e^{2\pi i v \omega}, \quad |v| < n$$

$$c) \hat{R}_z(v) = \frac{1}{Q} \sum_{k=1}^Q \hat{f}_y(\omega_k) e^{2\pi i v \omega_k}, \quad v = 0, 1, \dots, M,$$

where $\omega_k = (k-1)/Q$, $k = 1, \dots, Q$.

Implications: Before proving this theorem we note that parts (a) and (b) show that \hat{f}_z and \hat{R}_z are related in a very special way, as what are called Fourier transform pairs. Also, part (c) shows that the sample autocovariances can be calculated as the DFT of the sample spectral density (which is itself the square modulus of a DFT) of the padded series y . This is important because in some circumstances doing two DFTs (one to get \hat{f}_y and one to get \hat{R}_z) can

be much faster than calculating \hat{R}_z directly from its defining formula. We will discuss this further when we introduce the fast Fourier transform.

Proof of Theorem 1.4.4: To prove part (a) we note that

$$\begin{aligned} n\hat{f}_z(\omega) &= \left| \sum_{t=1}^n z(t)e^{2\pi i(t-1)\omega} \right|^2 \\ &= \left(\sum_{t=1}^n z(t)e^{2\pi i(t-1)\omega} \right) \left(\sum_{s=1}^n z(s)e^{-2\pi i(s-1)\omega} \right), \end{aligned}$$

since the complex conjugate of this first term in parentheses is in fact the second term. Thus,

$$n\hat{f}_z(\omega) = \sum_{t=1}^n \sum_{s=1}^n z(t)z(s)e^{-2\pi i(s-t)\omega}.$$

Instead of doing this double sum by letting t and s vary freely, we will add the terms for a fixed value of $s - t$ and then add the results for the various possible values of $v = s - t$. This is the discrete analog of change of variables in integration. Note that $s - t$ ranges from $1 - n$ to $n - 1$. There are $n - |v|$ terms in the double sum that have $s - t = v$, namely, the terms

$$z(1)z(1+|v|)e^{-2\pi i v \omega}, \dots, z(n-|v|)z(n)e^{-2\pi i v \omega}.$$

Thus,

$$\begin{aligned} \hat{f}_z(\omega) &= \frac{1}{n} \sum_{v=-(n-1)}^{n-1} \sum_{t=1}^{n-|v|} z(t)z(t+|v|)e^{-2\pi i v \omega} \\ &= \sum_{v=-(n-1)}^{n-1} \left(\frac{1}{n} \sum_{t=1}^{n-|v|} z(t)z(t+|v|) \right) e^{-2\pi i v \omega} \\ &= \sum_{v=-(n-1)}^{n-1} \hat{R}_z(v) e^{-2\pi i v \omega}, \end{aligned}$$

since $\bar{z} = 0$.

To prove part (b) we can now write

$$\begin{aligned} \int_0^1 \hat{f}_z(\omega) e^{2\pi i v \omega} d\omega &= \int_0^1 \left(\sum_{k=-(n-1)}^{n-1} \hat{R}_z(k) e^{-2\pi i k \omega} \right) e^{2\pi i v \omega} d\omega \\ &= \sum_{k=-(n-1)}^{n-1} \hat{R}_z(k) \int_0^1 e^{2\pi i (v-k)\omega} d\omega. \end{aligned}$$

But this integral is zero unless $k = v$ in which case it has value one and this fact gives part (b). For part (c), note that

$$\hat{R}_y(v) = \begin{cases} \hat{R}_z(v), & |v| < n \\ 0, & |v| \geq n, \end{cases}$$

and thus by part (a)

$$\begin{aligned} \frac{1}{Q} \sum_{k=1}^Q \hat{f}_y(\omega_k) e^{2\pi i v \omega_k} &= \frac{1}{Q} \sum_{k=1}^Q \left(\sum_{j=-(n-1)}^{n-1} \hat{R}_z(j) e^{-2\pi i j \omega_k} \right) e^{2\pi i v \omega_k} \\ &= \sum_{j=-(n-1)}^{n-1} \hat{R}_z(j) \frac{1}{Q} \sum_{k=1}^Q \left(e^{2\pi i (v-j)/Q} \right)^{k-1}, \end{aligned}$$

and this last sum is again, as in the proof of Theorem 1.4.1, a geometric series with $r = e^{2\pi i (v-j)/Q}$. Again, this sum is zero unless $(v-j)/Q$ is an integer, in which case it has the value Q . But the only way that $v-j$ can be a multiple of Q is if $v = j$, since the most that v and j can differ is $Q-1$ because $-(Q-n) \leq v \leq Q-n$ and $-(n-1) \leq j \leq n-1$. Thus the only term in the sum over j that is nonzero is when $j = v$, which is in fact $\hat{R}_z(v)$, which was to be shown.

The Fast Fourier Transform (FFT)

To calculate the discrete Fourier transform

$$w(k) = \sum_{t=1}^n x(t) e^{2\pi i (t-1)(k-1)/n}, \quad k = 1, \dots, n,$$

of data $x(1), \dots, x(n)$ appears to require n^2 multiplications and additions (n for each of the n values of k) and many evaluations of complex exponentials. In the mid-1960s various researchers made use of a variety of trigonometric identities to obtain algorithms called fast Fourier transform (FFT) algorithms (see Brigham (1974)) that essentially require only $n(p_1 + \dots + p_k)$ multiplications and additions, where p_1, \dots, p_k are the prime factors of n , and a greatly reduced number of evaluations of complex exponentials. For example, if $n = 1024 = 2^{10}$, then the number of operations is $1024(2 + \dots + 2) = 10 \cdot 2 \cdot 1024 = 2 \log_2 1024$, which is approximately 50,000 as opposed to $1024^2 \sim 1,000,000$, a savings of a factor of about 50. In Section A.3 we describe some of the details of how the FFT works.

Note that if n is not very composite, that is, it has some large prime factors, the FFT saves very little over a straightforward DFT. The subroutine that is used in TIMESLAB for calculating the FFT requires that n meets two

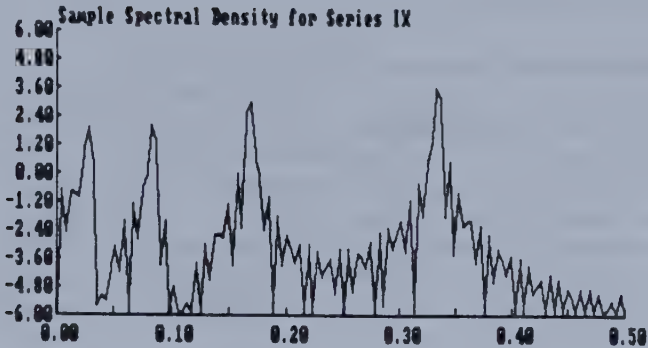


Figure 1.8. Sample Spectral Density of Series IX at 256 Frequencies.

requirements. First, the largest prime factor of n must be less than or equal to 23. Second, the product of what are called the “square-free” prime factors of n must be less than or equal to 210. For example, the product of the square-free prime factors of $n = 1260 = 2^2 \cdot 3^2 \cdot 5 \cdot 7$ is $5 \cdot 7 = 35$, which meets the requirements. If the user tries to use a command that asks **TIMESLAB** to find the DFT of length n and n does not meet the above requirements, then a message is displayed on the screen and the command aborts. Sometimes a user can write a macro that performs the same task as a command but calls the **FFT** command explicitly. This would allow the task to be performed even though it may take a very long time. In almost all cases padding with zeros to increase n is sufficient. Note that Series I ($n = 365$), II ($n = 370$), and III and IV ($n = 106$) each had to be padded with zeros before finding their Fourier transform, and thus Figure 1.6 actually contains graphs of the sample spectral density for those series. In most cases, the two graphs (periodogram and sample spectral density) will look very similar, that is, the features of the periodogram (excess of low or high frequencies or existence of peaks) will also be present in the sample spectral density. To illustrate this, consider Figure 1.8, which is the sample spectral density of Series IX, the sum of four cosine curves. The four curves were designed to have periods that all divide $n = 144$, and thus the amplitudes of all frequency components in the DFT for 144 frequencies are zero except for frequencies $1/36$, $1/12$, $1/6$, and $1/3$. However, if we were to find the DFT for $Q = 256$, as in Figure 1.8, none of the amplitudes would be zero, and the amplitudes for the important frequencies would not be calculated. Note that there are still four sharp peaks in the sample spectral density.

The FFT Command

The TIMESLAB command

FFT(zr,zi,n,m,sign,wr,wi)

calculates the DFT (if **sign**=1) or inverse DFT (if **sign**= -1) of the **n** complex numbers having real and imaginary parts **zr** and **zi** and stores the real and imaginary parts of the first **m** values of the result in the arrays **wr** and **wi**. We have up to now considered only Fourier transforms where the input is real. We will have occasion later to need the transform of complex arrays as well. The command

FFT(z,n,m,sign,wr,wi)

is the same as the form above except that **zi** is taken to be zero.

If **n** is not composite enough, the **FFT** command will display a message stating that the FFT algorithm is not available, and will use the defining formula to calculate the transform. A message is displayed warning that the command may take a long time to execute.

The FFT and Convolutions

Let $z(1), \dots, z(n)$ be a data set such that $\bar{z} = 0$. The FFT is useful in calculating the autocovariance function \hat{R}_z , which is defined to be the convolution

$$\hat{R}_z(v) = \frac{1}{n} \sum_{t=1}^{n-|v|} z(t)z(t+|v|), \quad |v| < n.$$

If we use this formula to find $\hat{R}_z(0), \dots, \hat{R}_z(M)$, we need

$$n + (n-1) + \dots + (n-M) = (M+1)n - \frac{M(M+1)}{2}$$

multiplications and additions. If, on the other hand, we pad z with $r \geq M$ zeros such that $Q = n+r$ is the next power of 2 greater than or equal to $n+M$, then we can use part (c) of Theorem 1.4.4 to get the desired quantities by doing two FFTs of length Q , that is, in $2Q \log_2(Q)$ multiplications and additions plus some evaluations of complex exponentials. As an example of this, suppose $n = 1000$ and $M = 100$. Then if $Q = 2048$, we would require $2(2048)(11)=45,056$ operations to do the two FFTs versus $101,000-5,050=95,950$ operations using the convolution formula. In Example 1.9 we give a macro that allows one to experiment with which method is faster for various values of n and M .

The CORR Command

The CORR command is flexible in that it can calculate just correlations, just the sample spectral density, or both at the same time. If both are asked for, CORR uses the two FFT method for finding correlations and does the padding with zeros automatically. The input data array is not affected in any way by the command. Thus for $Q \geq n + M$ and $M > 0$,

`rho=CORR(x,n,M,Q,iopt,R0,f)`

will calculate $R0 = \frac{1}{n} \sum_{t=1}^n z^2(t)$ and

$$\rho(v) = \frac{\sum_{t=1}^{n-v} z(t)z(t+v)}{R0}, \quad v = 1, \dots, M,$$

and

$$f(j) = \frac{1}{n} \left| \sum_{t=1}^n z(t) e^{2\pi i(t-1)(j-1)/Q} \right|^2, \quad j = 1, \dots, [Q/2] + 1,$$

where $z(t) = x(t)$ if `iopt=2` or $z(t) = x(t) - \bar{x}$ if `iopt=1`.

To get the first M correlations and the periodogram (as opposed to the sample spectral density function above), CORR must be called twice:

`rho=CORR(x,n,0,n,iopt,R0,per)`

to get the periodogram, and

`rho=CORR(x,n,M,0,iopt,R0,per)`

to get the correlations. To get $R0$ and nothing else, one can use

`rho=CORR(x,n,0,0,iopt,R0,per).`

If one or more of the output arguments is not calculated, then no such variable is formed by TIMESLAB but some name must be placed in the argument list as a "place-holder."

1.5. Transformations

Some of the time series analysis techniques that we will introduce in later chapters assume (1) that the data being analyzed have no deterministic trends or cycles, and (2) that the variability in the data is constant over time. The traditional method of analyzing data that fail to meet these requirements is to do the analysis in three steps: (1) try various transformations until the result

appears to meet the requirements, (2) analyze the result of step 1, and (3) do the inverse operation of what was done in step 1. This strategy is very similar to that used in regression analysis. In this section we will describe some of the transformations that are often used in time series analysis.

1.5.1. Stabilizing Variance

Suppose that the variability in a data set x appears to be increasing as time increases (see the airline data for example). If the mean level of the data is also increasing with time, then the variability in $y(t) = \log x(t)$ should appear fairly constant over time. This is a fairly common occurrence in real data. In general, one might try various power transformations to obtain a series having constant variability; that is,

$$y(t) = (x(t))^r.$$

Note that in TIMESLAB it is easy to raise each element of an array to a power r by using the Type 2 command $y=x^r$, and that the command $y=\text{LOGE}(x,n)$ will find the natural log of an array. Also, for a positive number c ,

$$\log_{10} c = \frac{\log_e c}{\log_e 10},$$

and so we can find the base 10 logarithm of an array x by

```
y=LOGE(x,n)
ee=LOGE(10)
y=y/ee
```

1.5.2. Removing Trends

Another common phenomenon in time series data is that the values appear to be growing in some polynomial fashion with time, particularly linearly; that is, the data appear to follow

$$x(t) = a + bt + \epsilon(t),$$

where $\epsilon(t)$ is white noise. Such polynomial trends are often removed by using differencing.

Definition. The d th difference of a time series data set x having n elements is a data set y having $n - d$ elements obtained by

$$y(t) = x(t + d) - x(t), \quad t = 1, \dots, n - d.$$

To illustrate the use of differencing, suppose that $x(t) = a + bt + \epsilon(t)$. Then

$$\begin{aligned} y(t) &= x(t + 1) - x(t) = a + bt + \epsilon(t + 1) - [a + b(t - 1) + \epsilon(t)] \\ &= b + \epsilon(t + 1) - \epsilon(t), \end{aligned}$$

which no longer has a linear trend. It is easy to see that if x contains a general d th-degree polynomial trend, applying first differencing d times will remove it, while if x has a cycle of length s time units, then taking s th differences will remove the cycle.

Perhaps the most well known example of differencing is the airline data that we have been analyzing. The variability in the data is increasing with time and thus a log transform is usually applied. The result of this transform has an obvious 12-month cycle and also a linear trend. Thus the traditional advice on a series such as this is to take the first difference of the 12th difference of the log of the original data.

The DIFF Command

The command

```
y=DIFF(n,d,x)
```

will form an output array y of length $n - d$ as the d th difference of x . Thus the commands

```
y=LOGE(x,n)
y=DIFF(n,1,y)
nm1=n-1
y=DIFF(nm1,12,y)
```

will form the first and 12th difference of the log of an array x . Note that the resulting array will be of length $n - 13$.

Applying differencing is very simple and often leads to good results, but some care must be taken. For example, in our linear trend plus white noise example given above, the result of taking first differences was

$$y(t) = b + \epsilon(t + 1) - \epsilon(t),$$

which is a special kind of time series called a moving average time series (see Section 2.5.3) of order one with coefficient -1 . Some standard analysis techniques cannot be applied to such a time series.

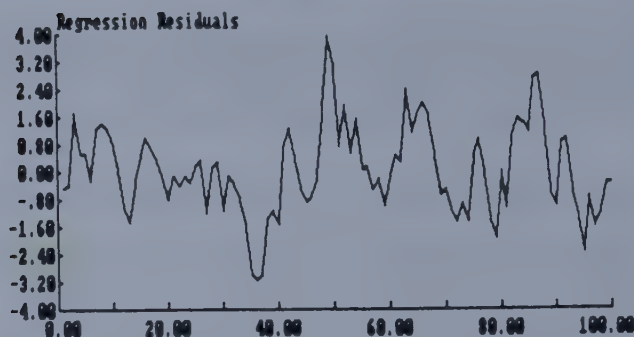


Figure 1.9. Residuals from Regression for Series V.

Regression Analysis

A traditional method of removing trends and cycles from time series data is to do ordinary regression of the data on the deterministic functions of time. For example, Series V of the univariate series that we have been analyzing was formed by

$$x(t) = 10 + .1t + 3 \cos \frac{2\pi(t-1)}{10} + \epsilon(t), \quad t = 1, \dots, 100,$$

where $\epsilon(t)$ is a series having serial correlation (actually $\epsilon(t)$ is a realization from an AR(1) process with coefficient $-.75$; see Section 2.5.4). If we did ordinary least squares regression of the form

$$y(t) = \beta_0 + \beta_1 t + \beta_2 \cos \frac{2\pi(t-1)}{10} + \epsilon(t),$$

we would obtain the residuals given in Figure 1.9, which can then be further analyzed by time series analysis. See Example 1.10 for the macro that generated this figure, and Section A.5 for a complete description of the types of regression analyses that are available in TIMESLAB.

1.5.3. Accounting for Seasonal Variability

Often it is of interest in seasonal time series to analyze how a data set differs from regular seasonal variation. For example, suppose that x consists of m years of monthly data, that is, $n = 12m$. We can define the monthly means and variances to be the means and variances of each of the 12 data sets

consisting of like months; the Januaries, Februaries, and so on. Thus,

$$\bar{x}_k = \frac{1}{m} \sum_{t=1}^m x(k + 12(t-1))$$

$$s_k^2 = \frac{1}{m} \sum_{t=1}^m [x(k + 12(t-1)) - \bar{x}_k]^2,$$

$k = 1, \dots, 12$. Note that other seasonal (quarterly or hourly, for example) means and variances can be defined similarly. See Problem T1.7 for more information about seasonal means.

The SUBMNS and DIVSDS Commands

The SUBMNS command will calculate and/or subtract seasonal means from a data set. It can also be used to add back some seasonal means that were previously subtracted. This command is also used to just calculate and/or subtract the “overall” mean of a data set, that is, the mean of the entire data set. Thus,

y=SUBMNS(x,n,d,xbar)

will calculate and subtract seasonal means **xbar** of seasonality **d** for an array **x** and return the result in **y**. The form

y=SUBMNS(x,n,d,xbar,iopt)

with **iopt=1** is the same as that above, while if **iopt=0**, then **xbar** is formed but **y** is not, and if **iopt=2**, then an inputted array **xbar** is added to **x** to get **y**.

The command DIVSDS operates the same as SUBMNS except that it refers to calculating and/or dividing or multiplying seasonal standard deviations, that is, to square roots of the seasonal variances defined above. Again, if **d=1** in any of the forms above, then the command operates on the overall mean or standard deviation.

1.5.4. General Smoothing Operations

Another approach to removing deterministic-looking parts of a time series is to use general methods of smoothing data and then analyze the deviations from the smooth version. For example, one way to smooth wiggly data is to

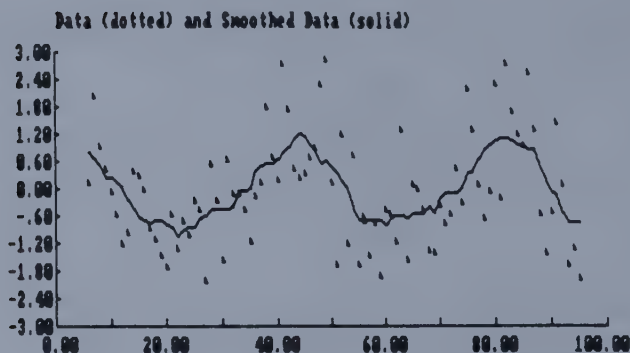


Figure 1.10. An Example of the Moving Average Smoother.

use a moving average smoother, for example, the one of length 3:

$$y(1) = \frac{x(1) + x(2) + x(3)}{3}$$

$$y(2) = \frac{x(2) + x(3) + x(4)}{3}$$

$$\vdots$$

$$y(n-2) = \frac{x(n-2) + x(n-1) + x(n)}{3}.$$

Since consecutive y 's have two of the x 's in common in their average, we would expect that the y 's won't vary as much as the original x 's; that is, y will be much smoother in appearance. In Figure 1.10 we have generated 100 points of a cosine of amplitude one and period 40, added $N(0,1)$ white noise to it, and then used a moving average smoother of length 11. Note that the last smoothed value corresponds to the 95th data point, and so we have superimposed the 6th through 95th data points and the smoothed data. Note that the smoothed data clearly exhibit the cosine curve, which is not obvious in the original data. See Example 1.11 for the macro that formed this figure.

The FILT Command

The moving average smoother is a special case of what is called a linear filter:

$$y(t) = \beta_0 x(t+m) + \beta_1 x(t+m-1) + \cdots + \beta_m x(t), \quad t = 1, \dots, n-m.$$

The command

```
y=FILT(x,beta,beta0,n,m)
```

will perform such a filtering operation where **beta** is an array containing β_1, \dots, β_m , and **beta0**= β_0 . We will discuss linear filters in detail in Section 2.3.

1.6. Some Simple Forecasting Methods

An important objective of time series analysis in many scientific areas is to forecast (predict) future values of a time series $x(1), \dots, x(n)$. In this section we begin our discussion of forecasting methods by looking at some simple methods. In Section 2.4 we study the theory of prediction, and in Chapter 3 we discuss some of the more elaborate methods that are used in practice.

Each of the methods in this section can be viewed as having two parts: (1) a model-fitting part, and (2) a forecasting part. Note that many statistical procedures are of this form.

1.6.1. The Inverse Differencing Method

A simple but effective method for extrapolating trends and cycles in data that can be handled by differencing is to extend the data set with M new values in such a way that if the extended series of length $n + M$ were then differenced, the last M values (the ones corresponding to the new values) would all equal the mean of the differences of the original data. For example, if a data set appears to have a linear trend, we might detrend it by applying first differences. We could then find a new data point $x(n + 1)$ so that $x(n + 1) - x(n) = \bar{z}$, where \bar{z} is the mean of the differences of $x(1), \dots, x(n)$. Thus $x(n + 1) = x(n) + \bar{z}$, and if we continue this process, we obtain

$$x(n + h) = x(n + h - 1) + \bar{z}, \quad h \geq 1.$$

If we have data that appear to contain both a linear trend and a seasonal cycle, we might apply both first and d th differences, where d is the length of the seasonal cycle. For monthly data having an annual cycle, we would solve

$$x(n + 1) - x(n) - x(n - 11) + x(n - 12) = \bar{z},$$

where \bar{z} is the mean of the first and 12th differences of $x(1), \dots, x(n)$. This results in the predictor

$$x(n + h) = x(n + h - 1) + x(n + h - 12) - x(n + h - 13) + \bar{z}, \quad h \geq 1.$$

The EXTEND Command

Given an array x of length n containing a data set, an integer $d1$, and optionally another integer $d2$, the command

```
y=EXTEND(x,n,next,d1[,d2])
```

will carry out the process of extending an array by the inverse operation of differencing as described above. The first n elements of the array x will be extended by $next$ points and the result will be placed in the array y which will be of length $n+next$. One or two differences can be specified depending on whether or not the argument $d2$ is included.

To illustrate the use of the **EXTEND** command, consider Figure 1.11, which contains the result of the commands

```
READ(air,x,n)
y=LOGE(x,n)
yext=EXTEND(y,n,24,1,12)
np24=n+24
yext=EXP(yext,np24)
LABEL(yext)'Airline Data and Extended Values'
LABEL(x)' '
PLOT2(x,yext,n,np24,2,1,0,170,0,800)
```

The observed data values are represented on the plot by both a solid curve and an \times at each point, while the extended values are represented only by the solid curve. The extended values are the last two cycles on the graph and are consistent with what we would expect from a predictor. It is not surprising that we could use this “naive” method on the airline data as it is a long-memory series and we would expect that almost any method would work well, including just drawing in the next two cycles by hand. For series that do not follow such a deterministic pattern we will have to use more sophisticated methods.

1.6.2. The Regression Method

If the data appear to follow a deterministic function of time that can be expressed as a simple or multiple linear regression model, we can predict future values of the series by substituting future values of t into the least squares regression function (see Section A.5 for a discussion of multiple linear regression). For example, if

$$x(t) = a + bt + c \cos \frac{2\pi(t-1)}{p} + \epsilon(t),$$

for $t = 1, \dots, n$, where the period p is known, we can estimate a , b , and c by their least squares estimates \hat{a} , \hat{b} , and \hat{c} , and forecast a value h steps into the

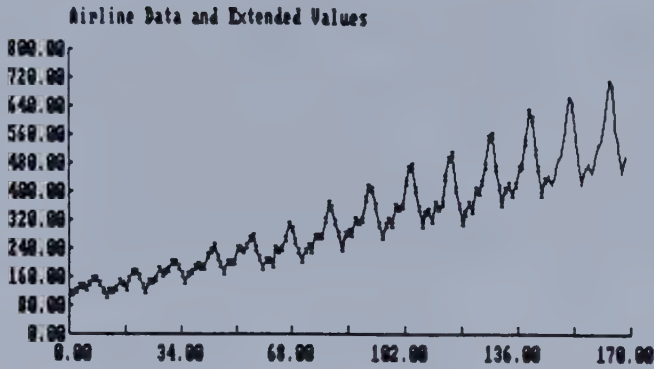


Figure 1.11. Airline Data and Extended Values.

future by

$$\hat{x}(n+h) = \hat{a} + \hat{b}(n+h) + \hat{c} \cos \frac{2\pi(n+h-1)}{p}.$$

1.6.3. Simple Moving Average

This method models an observation as a simple average of the previous m observations where m is to be chosen. For a given moving average length k we can calculate

$$S(k) = \frac{1}{n-k} \sum_{t=k+1}^n \left(x(t) - \frac{1}{k} \sum_{j=1}^k x(t-j) \right)^2$$

as a measure of how well the simple moving average model of length k fits the observed data. We chose m as the value of k minimizing $S(k)$. Then we forecast future values recursively. For example, if $m = 3$ we calculate

$$\hat{x}(n+1) = \frac{1}{3} [x(n) + x(n-1) + x(n-2)]$$

$$\hat{x}(n+2) = \frac{1}{3} [\hat{x}(n+1) + x(n) + x(n-1)]$$

$$\hat{x}(n+3) = \frac{1}{3} [\hat{x}(n+2) + \hat{x}(n+1) + x(n)],$$

and so on.

1.6.4. Simple Exponential Smoothing

Instead of modeling an observation as a simple average of the previous m observations, exponential smoothing methods model $x(t)$ as a weighted average of all of the previous values. The type of weights that are used depends on the appearance of the data and leads to methods having a variety of names. We will discuss only the simple exponential smoothing technique which is most suitable for data that appear to have no linear or seasonal trends.

The simple exponential smoothing method consists of modeling $x(t)$ as a weighted average:

$$\hat{x}(t+1) = \sum_{j=1}^t \beta_j x(t+1-j), \quad t \geq 1.$$

If we let $\beta_j = \alpha(1-\alpha)^{j-1}$, where $0 \leq \alpha \leq 1$; that is, we let the weights decay exponentially to zero, then $\sum_{j=1}^{\infty} \beta_j = 1$, and thus for large t , the weights will approximately sum to one. We can choose α as the value of a minimizing

$$S(a) = \sum_{t=2}^n \left(x(t) - \sum_{j=1}^{t-1} a(1-a)^{j-1} x(t-j) \right)^2,$$

and then forecast the value at time $n+1$ by

$$\hat{x}(n+1) = \sum_{j=1}^n \alpha(1-\alpha)^{j-1} x(n+1-j).$$

The calculations involved in this process can be greatly reduced by noting that

$$\hat{x}(t+1) = \alpha x(t) + (1-\alpha)\hat{x}(t),$$

where $\hat{x}(1)$ is defined to be $x(1)$. We can also write this as $\hat{x}(t+1) - (1-\alpha)\hat{x}(t) = \alpha x(t)$, which is called a difference equation of order one. Difference equations are very important in the study of time series analysis.

1.6.5. Difference Equations and the Behavior of Forecasts

Definition. Let $z(\cdot)$ and $w(\cdot)$ be sequences of real numbers. Then

$$z(t) + \alpha_1 z(t-1) + \cdots + \alpha_p z(t-p) = w(t)$$

is called a difference equation of order p , coefficients $\alpha_1, \dots, \alpha_p$, and forcing term $w(\cdot)$.

To calculate the values of z for all values of t it is sufficient to know all of the values of w and any p consecutive values of z . These p values are

called starting values or initial conditions. If we know the starting values $z(1), \dots, z(p)$ and the values $w(p+1), \dots, w(n)$ for a difference equation, then we can find $z(p+1), \dots, z(n)$ recursively by

$$z(p+j) = w(p+j) - \sum_{k=1}^p \alpha_k z(p+j-k), \quad j = 1, \dots, n-p.$$

Note that $w(p+1)$ and $z(1), \dots, z(p)$ are used to find $z(p+1)$, which is in turn used in finding $z(p+2)$, and so on.

The ARDT Command

In the command

`z=ARDT(alpha,p,n,w)`

the array `alpha` contains $\alpha_1, \dots, \alpha_p$, while the array `w` has $z(1), \dots, z(p)$ as its first p elements, and $w(p+1), \dots, w(n)$ as its last $n-p$ elements. The output array `z` contains $z(1), \dots, z(n)$. Thus in

$$\hat{x}(t) - (1-a)\hat{x}(t-1) = ax(t-1)$$

we have $p = 1$, $z(t) = \hat{x}(t)$, $z(1) = 0$, $\alpha_1 = -(1-a) = a-1$, and $w(t) = ax(t-1)$. Thus we can find $\hat{x}(1), \dots, \hat{x}(n)$ and $S(a) = \sum_{t=2}^n (x(t) - \hat{x}(t))^2$ by

```
oa=a-1
alpha=<oa>
w=a*x
w=<x[1],w>
xhat=ARDT(alpha,1,n,w)
e=x-xhat
Sa=DOT(e,e,n)
```

In Example 1.12 we give a macro which will choose α and superimpose the plots of the data and the resulting fitted values. Note that we can also write the simple exponential smoothing recursion as

$$\hat{x}(t+1) = \hat{x}(t) + \alpha(x(t) - \hat{x}(t));$$

that is, the predicted value at time $t+1$ is that for time t plus an adjustment for the error in using $\hat{x}(t)$ to model $x(t)$. To forecast $x(n+h)$ for $h > 1$, some authors suggest using

$$\hat{x}(n+h) = \alpha x(n) + (1-\alpha)\hat{x}(n+h-1).$$

Many of the forecasting methods used in time series analysis produce forecasts that are future values of a difference equation. In Theorem 1.6.1 we present results that can be used to describe analytically the behavior of these forecasts. The results of this theorem will be useful in other contexts as well.

Definition. Let

$$\sum_{j=0}^p \alpha_j z(t-j) = w(t)$$

be a p th-order difference equation. If $w(t) = 0$ for all t , then the difference equation is said to be homogeneous. The polynomials

$$g(z) = \sum_{j=0}^p \alpha_j z^j \quad \text{and} \quad h(z) = \sum_{j=0}^p \alpha_j z^{p-j}$$

are called the characteristic and indicial polynomials, respectively, of the difference equation.

Let z_1, \dots, z_p be the zeros of h ; that is, $h(z_i) = 0$. Note that the zeros of g are the reciprocals of the zeros of h . These zeros play an important role in solving difference equations.

Theorem 1.6.1

SOLUTION OF DIFFERENCE EQUATIONS

Suppose that $z(t)$ satisfies a p th-order homogeneous difference equation, and let z_1, \dots, z_p be the zeros of the indicial polynomial of the equation. Then

a) The zeros are either real or occur in complex conjugate pairs; that is, if $z_i = a + bi$ is a complex valued zero, then $\bar{z}_i = a - bi$ is also a zero.

b) $z(t)$ is equal to the sum of terms of two types. If a real root z occurs m times, then a term of the form

$$(\beta_1 + \beta_2 t + \dots + \beta_m t^{m-1}) z^t$$

is included. For a distinct real root this term is $\beta_1 z^t$. If a conjugate pair ($z = a + bi$, $\bar{z} = a - bi$) occurs m times, then a term of the form

$$[\gamma_1 \cos(t\theta + \delta_1) + \gamma_2 t \cos(t\theta + \delta_2) + \dots + \gamma_m t^{m-1} \cos(t\theta + \delta_m)] r^t$$

is included, where r and θ are determined by expressing z and \bar{z} as $r(\cos \theta \pm i \sin \theta)$; that is, $r = \sqrt{a^2 + b^2}$ and $\theta = \arctan(b/a)$. In either case, the β 's,

γ 's, and δ 's are determined by solving the system of equations resulting from p initial conditions, that is, by setting any p known values of z equal to the sum of the terms described above.

c) If the zeros are all real and distinct, then as $t \rightarrow \infty$, the values of $z(t)$ either converge to zero, oscillate between $-\infty$ and ∞ , or diverge to ∞ .

Note that part (c) is just one possible application of part (b) (see Problem T1.11 for another). To illustrate solving a difference equation analytically, consider

$$z(t) - z(t-1) + z(t-2) - z(t-3) = 0, \quad t \geq 4,$$

with initial conditions $z(1) = 1$, $z(2) = -1$, and $z(3) = 1$. Then the indicial polynomial is

$$h(z) = z^3 - z^2 + z - 1 = (z^2 + 1)(z - 1),$$

which has zeros $z_1 = i$, $z_2 = -i$, and $z_3 = 1$. For z_3 , a term of the form $\beta 1^t = \beta$ is included, while for z_1 and z_2 , we have $r = 1$ and $\theta = \pi/2$, and thus a term of the form $\gamma \cos(t\pi/2 + \delta)$ is included. Thus,

$$z(t) = \beta + \gamma \cos\left(\frac{t\pi}{2} + \delta\right),$$

where β , γ , and δ are found by solving the equations

$$z(1) = 1 = \beta + \gamma \cos\left(\frac{\pi}{2} + \delta\right)$$

$$z(2) = -1 = \beta + \gamma \cos(\pi + \delta)$$

$$z(3) = 1 = \beta + \gamma \cos\left(\frac{3\pi}{2} + \delta\right).$$

These equations are satisfied by $\beta = 1$, $\gamma = -2$, and $\delta = \pi$. This gives

$$z(t) = 1 - 2\cos\left(\frac{t\pi}{2} + \pi\right),$$

and thus z repeats its values every four points, with $z(4)$ through $z(7)$ given by 3, 1, -1 , and 1. We can verify this numerically by the commands

```
alpha=<-1,1,-1>
f=LINE(100,0,0)
f=<1,-1,1,f>
x=ARDT(alpha,2,100,f)
LIST(x,100)
```

If we were to plot the real and imaginary parts of a complex number $z = a + bi$ on an X - Y plane, then $|z| = \sqrt{a^2 + b^2}$ is greater than one if and

only if the plotted point falls outside of a circle of radius one that is centered at the origin. This circle is called the unit circle, and we will henceforth refer to zeros being inside, outside, or on the unit circle. In Section 2.6 we will describe methods for determining whether the zeros of a polynomial are outside of the unit circle without actually finding them.

Commands for Polynomials

If **alpha** and **beta** are arrays of length **p** and **q** containing the coefficients of polynomials of degrees **p** and **q** whose zero degree coefficients are one, then the command

```
gamma=INVPOLY(alpha,q,M)
```

will find the first **M** coefficients of the reciprocal polynomial corresponding to that having coefficients **alpha**, while the command

```
gamma=MULTPOLY(alpha,beta,p,q)
```

will multiply the polynomials of degree **p** and **q** to find the coefficients **gamma** of the resulting polynomial of degree **p + q**.

The commands **POLYROOTS** and **ROOTSPOLY** will find the roots of a polynomial given its coefficients and find the coefficients given the roots, respectively:

```
POLYROOTS(a,a0,deg,maxit,rreal,rimag,ier)
ROOTSPOLY(rreal,rimag,deg,a).
```

1.7. Describing the Distribution of Data

TIMESLAB has three commands for studying the marginal distribution of data. The simplest of these is the **HIST** command. **HIST** will produce a histogram of a data set using either a user-specified number of intervals (bins) or using the number of intervals chosen by

$$n_{\text{bins}} = \left\lceil \frac{r_x}{3.5s_x} n^{1/3} + 1 \right\rceil,$$

where r_x and s_x are the range and standard deviation of the data, respectively.

The second command one can use is **INFQNT**, which produces what Parzen (1983a) calls an informative quantile plot, that is, a plot of

$$y(i) = \frac{x_{(i)} - \tilde{m}}{2IQR} \text{ versus } u(i) = \frac{i - .5}{n}, \quad i = 1, \dots, n,$$

where \tilde{m} and IQR are the median and interquartile range of the data, $x_{(i)}$ is the i th smallest x , and the values plotted on the vertical axis are truncated

Table 1.3. Kernels in DENSITY

kernel	Kernel Name	$W(t)$
1	Rectangular	$\frac{1}{2}$
2	Cosine	$\frac{1}{2}(1 + \cos \pi t)$
3	Epanechnikov	$\frac{3}{4}(1 - t^2)$
4	Biweight	$\frac{15}{16}(1 - t^2)^2$
5	Triangular	$1 - t $
6	Gaussian	$1.012339e^{-\pi t^2}$
7	Parzen	$\begin{cases} \frac{4}{3} - 8t^2 + 8 t ^3, & t \leq .5 \\ \frac{8}{3}(1 - t)^3, & t > .5 \end{cases}$

to fall between -1 and 1 . Whenever an informative quantile plot is produced, a reference line is placed on the plot. This line is the true quantile plot of a $U(0, 1)$ random variable. One can get an idea of what the informative quantile plot should look like for various distributions by generating a sample via the **WN** command and then calling **INFQNT** (see Example 1.1).

The **DENSITY** Command

TIMESLAB has the command **DENSITY** to form what is called a kernel probability density estimator. This estimator is also called a nonparametric density estimator since it assumes no particular functional form for the density. The histogram of a data set is in fact a simple form of kernel density estimator. It divides the range of the data into nonoverlapping intervals and counts how many of the data values are in each of the intervals. Then a bar graph of relative frequencies (counts divided by the sample size) is formed. Thus we can think of the height of the bar for a particular interval as the average number (or proportion) of the total data that are in that interval. In a more general kernel density estimator, the range is also divided into intervals, but these intervals are allowed to overlap, and the aim is to estimate the density at the center point of each interval. Instead of just counting the number of points in the interval, the method assigns a number between zero and one to each point in the interval with a data point at the center of the interval getting the highest possible value. The farther from the center that a point is, the less weight it is given. The function that is used to determine these “weights” is called a kernel, and thus the method is called the kernel method. The intervals are determined by specifying their center point and their width; that is, the i th interval is

given by $[y(i) - h, y(i) + h]$, where the number h is called the "bandwidth." Notice that the width of an interval is not h but rather $2h$.

Formally, then, the density estimator at a point $y(i)$ is given by

$$\hat{f}(y(i)) = \frac{1}{nh} \sum_{j=1}^n W\left(\frac{y(i) - x(j)}{h}\right),$$

where the kernel W is a function that is nonnegative and symmetric about zero. Thus to form a kernel density estimate, one must specify: (1) the number of points at which to find the estimator, (2) the kernel to be used, and (3) the bandwidth to be used. In the TIMESLAB command

DENSITY(x,n,rbins,npts,kernel,y,fy)

the first two arguments are the data array and how many of the data points are to be used to find the estimator, **rbins** is a real scalar that determines the bandwidth to be used (see below), **npts** is an integer that determines the number of points at which the estimator is to be calculated, and **kernel** is an integer specifying which of seven available kernels the user wants (see Table 1.3). Note that each of the seven kernels in Table 1.3 is defined for $|t| \leq 1$ and is zero for $|t| > 1$. Each of them is in fact a probability density function on $[-1,1]$ as they are positive and integrate to one. Thus the resulting density estimators are comparable.

If **npts** is positive, then TIMESLAB will form the array **y**, of length **npts**, as the **npts** equally spaced points between the smallest and largest data points inclusive. If **npts** is negative, then the user enters in the array **y** the **-npts** values at which the density estimator is to be found. In either case, the **DENSITY** command will return in the array **fy** the estimator at the points in **y**.

If the user enters a positive value for **rbins**, then the bandwidth that is used is given by

$$h = \frac{r_x}{2\text{rbins}},$$

where r_x is the range of the data. Specifying the bandwidth in this indirect way means that the user need not know the range of the data prior to issuing the **DENSITY** command. The value of **rbins** is very similar to the value of **nbins** in the **HIST** command. A large value for **rbins** will lead to a small value of h , and thus the resulting estimator will appear wiggly, just as using a large value of **nbins** in **HIST** will lead to a wiggly histogram. Thus typical values for **rbins** are similar to typical values for **nbins**, namely, in the range of four through twelve (see Problem C1.24).

If the user enters a value of zero for **rbins**, then **DENSITY** will use

$$\text{rbins} = \frac{r_x n^q}{3.5 s_x},$$

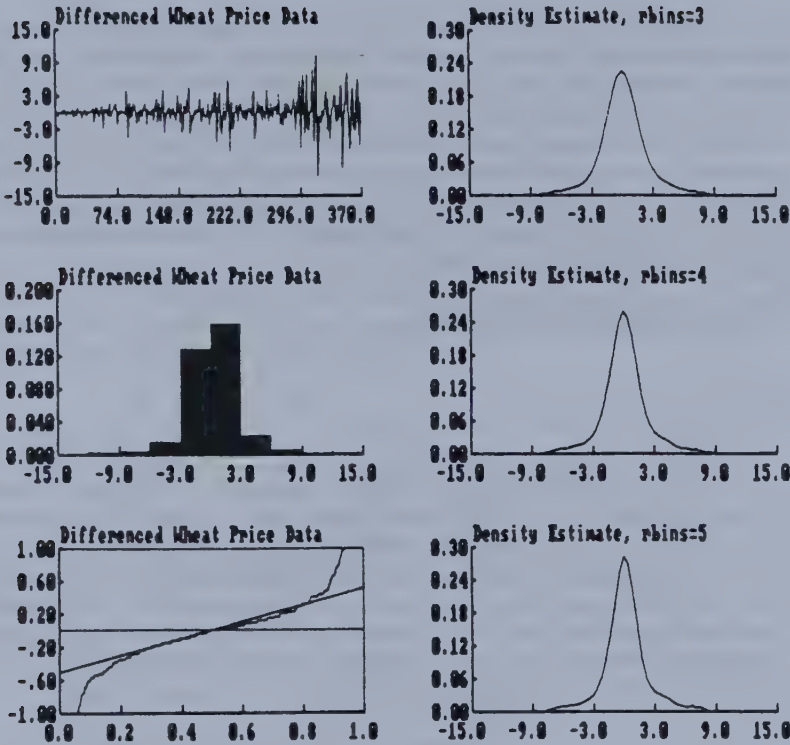


Figure 1.12. Studying the Distribution of the First Difference of the Wheat Price Data.

where s_x is the standard deviation of the data, and q is $1/3$ for the rectangular kernel and $1/5$ for the others. This translates into a bandwidth that is proportional to $n^{-1/3}$ for the rectangular kernel and $n^{-1/5}$ for the others.

Most users of kernel density estimators agree that the choice of kernel is not as important as the choice of h . Choosing an optimal bandwidth is a topic of much discussion in the literature (see Parzen (1962a), Tapia and Thompson (1978), and Silverman (1986)). We will see in Chapter 3 that basically the optimal bandwidth in the closely related problem of spectral density estimation is proportional to $n^{-1/5}$ for most kernels.

Unlike HIST and INFQNT, DENSITY does not produce a plot of the density that it calculates. To plot it one can use the PLOT command. Several such densities can be superimposed on the same axes using the PLOTK command. This allows the user to create informative labels and scales for the plot.

In Figure 1.12 we plot the first difference of the Beveridge wheat price data (the differenced values were divided by 10 so that the tic mark labels

on the vertical axis would be satisfactory), its histogram with the number of intervals chosen by the HIST command, its informative quantile plot, and three density estimates using the Parzen kernel for three values of `rbins`. Note that the informative quantile plot of a normally distributed sample should intersect the lower left and upper right corners of the plot. For the data that we are considering, the plot shows clearly the presence of too many large and small values for the data to be consistent with normality, even though the density estimates are clearly symmetric. See Example B.2 for another illustration of the use of the DENSITY command.

1.8. Examples and Problems

Example 1.1 WHITE NOISE DATA

The macro `WN.MAC` finds the data plot, histogram, and informative quantile plot for white noise series having eight different marginal distributions. The variables `n` (sample size) and `seed` must be defined prior to starting the macro. If the command `PSON` is issued prior to invoking the macro, then the three graphs for each of the eight series will also be printed (the `PRINTSEL` command has to have been issued also to tell `TIMESLAB` what type of printer is attached). Inserting the `PAGE` command after the `INFQNT` will also make it so that the three graphs for each distribution are on separate pages.

```

1 ;;
2 ;;   WN.MAC: macro to produce and plot white noise realizations
3 ;;           of length n from each of the 8 distributions in the
4 ;;           WN command.
5 ;;
6 ;;   INPUT: n, seed (random number generator seed)
7 ;;
8 ;;
9 PAUSE
10 ;start
11 x=WN(seed,10)           ;warm up generator
12 ntype=1                 ;initialize distribution counter
13 ;
14 ;startloop
15 ;
16 x=WN(0,n,ntype)         ;generate data
17 PLOT(x,n)               ;plot of time series
18 HIST(x,n)              ;histogram
19 INFQNT(x,n)             ;informative quantile plot
20 IF(ntype.eq.8,endoloop) ;check if this is last distribution
21 ntype=ntype+1           ;if not, update ntype and go back
22 GOTO(startloop)
23 ;

```



```

24 ;endloop
25 ;

```

Example 1.2 DESCRIPTIVE STATISTICS

The macro DESCRIBE.MAC calculates and displays the basic descriptive statistics introduced in Chapter 1 for a univariate time series. If the macro were executed after having defined $n=100$, $x=WN(12345,n)$, $M=20$, and $Q=n$, and we told TIMESLAB to obtain screen dumps of each of the graphs involved (from the graphics menu or by calling PSON prior to calling the macro), we would get the plots in Figure 1.13.

```

1 ;;
2 ;;   DESCRIBE.MAC: macro to find and display descriptive statistics
3 ;;                       for a univariate time series.
4 ;;
5 ;;   INPUT: x, n (data and sample size)
6 ;;           M (number of correlations), Q (number of frequencies
7 ;;           in [0,1]).
8 ;;
9 PAUSE
10 ;start
11 PLOT(x,n)                ;labels won't be nice
12 rho=CORR(x,n,M,0,1,r0,f) ;find correlations
13 PLOT(rho,M,0,M,-1,1)     ;plot them
14 rho=CORR(x,n,0,Q,1,r0,f) ;find sample spectral density
15 PLOTSP(f,Q,r0)           ;plot it
16 xx=SUBMNS(x,n,1,xbar)
17 part=PARCORR(xx,n,M,rvar) ;find partials and res. vars.
18 rvar=rvar/r0             ;standardize residual variance
19 LABEL(part)='Partials (solid)' ;prepare labels
20 LABEL(rvar)='Residual Variance (xxxx)' ;
21 PLOT2(part,rvar,M,M,1,2,0,M,-1,1) ;plot them

```

Example 1.3 SCATTERPLOTS

The macro SCATTER.MAC was used to produce the scatterplots in Figure 1.2. The plot produced by the last line of the macro does not have nice tic mark labels. Thus we ran the macro, inspected the resulting plot, and then called PLOT again except using the seven-argument form so that we would get pleasing tic mark labels. We also changed the label that is placed above the plot.

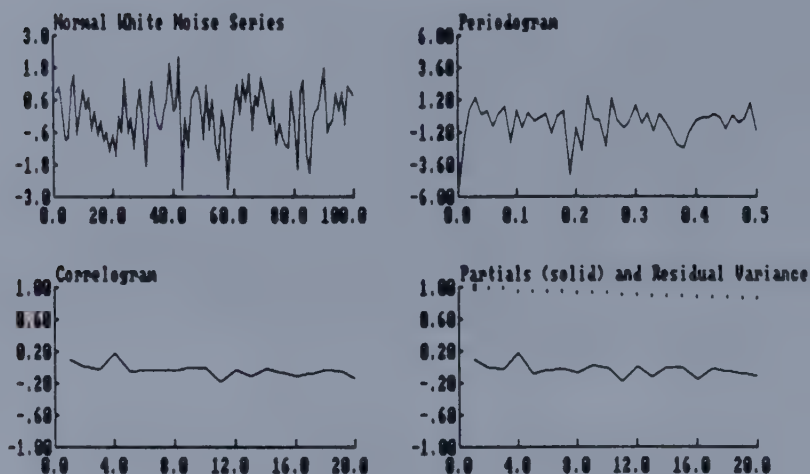


Figure 1.13. Example Output from DESCRIBE.MAC.

```

1 ;;
2 ;;   SCATTER.MAC: macro to produce the scatterplot of a data set
3 ;;           x (which is of length n) with itself for lag v.
4 ;;
5 ;;   INPUT: x, n, v
6 ;;
7 PAUSE
8 ;start
9 nmv=n-v
10 vpi=v+1
11 xlag=EXTRACT(x,vpi,n) ;this is how to lag a series
12 xx=x
13 LABEL(xlag)='Scatterplot for Lag #v#'
14 LABEL(xx)=' '
15 mnmv=-nmv ;using a negative number of points results in scatterplot
16 PLOT(xx,xlag,mnmv)

```

Example 1.4 SUM OF FOUR COSINES

The macro COS.MAC generates four cosine curves, plots each of them and their sum, and then calculates, plots, and lists the periodogram of the sum. The periodogram is zero except for the values at the frequencies of the individual cosine curves.

```

1 ;;
2 ;;   COS.MAC: macro to illustrate how the periodogram can
3 ;;       decompose a series into a sum of sinusoids.
4 ;;
5 ;;   INPUT: none
6 ;;
7 PAUSE
8 ;start
9 PLOTOM          ;stay in graphics mode
10 n=144
11 p=<12,36,6,3> ;periods of the four sinusoids
12 a=<2,1,6,7>   ;cosine coefficients
13 b=<3,3,1,4>   ;sine coefficients
14 ;
15 ;   Form individual sinusoids and the sum :
16 ;
17 x1=COS(n,a[1],p[1])
18 y1=SIN(n,b[1],p[1])
19 z1=x1+y1
20 x2=COS(n,a[2],p[2])
21 y2=SIN(n,b[2],p[2])
22 z2=x2+y2
23 x3=COS(n,a[3],p[3])
24 y3=SIN(n,b[3],p[3])
25 z3=x3+y3
26 x4=COS(n,a[4],p[4])
27 y4=SIN(n,b[4],p[4])
28 z4=x4+y4
29 z=z1+z2+z3+z4
30 CLEAN(x1,y1,x2,y2,x3,y3,x4,y4)
31 ;
32 ;   Use MAXMIN to find max and min so all 5 plots will
33 ;   be on the same scale:
34 ;
35 zz=<z,z1,z2,z3,z4>
36 MAXMIN(zz,n,xmax,imax,xmin,imin)
37 CLEAN(zz)
38 ;
39 ;   Form plotting labels :
40 ;
41 LABEL(z1)='Period 12 Amplitude Squared 13'
42 LABEL(z2)='Period 36 Amplitude Squared 10'
43 LABEL(z3)='Period 6 Amplitude Squared 37'
44 LABEL(z4)='Period 3 Amplitude Squared 65'
45 LABEL(z)='Sum of four cosine curves'
46 ;
47 ;   Plot :
48 ;
49 BATCHOM          ;no pauses between plots
50 PLOTSIZE(1)       ;upper left
51 PLOT(z1,n,0,n,xmin,xmax)
52 PLOTSIZE(2)       ;lower left
53 PLOT(z2,n,0,n,xmin,xmax)
54 PLOTSIZE(3)       ;upper right

```

```

55 PLOT(z3,n,0,n,xmin,xmax)
56 PLOTSIZE(4)                ;lower right
57 BATCHOFF                   ;do want a pause after 4th plot
58 PLOT(z4,n,0,n,xmin,xmax)
59 CLS                        ;clear screen and get ready for next 2 plots
60 PLOTSIZE(8)                ;bottom half
61 BATCHON                     ;no pause between the 2 plots
62 PLOT(z,n,0,n,xmin,xmax)
63 ;
64 ;   Calculate and display periodogram illustrating what
65 ;   periodogram means :
66 ;
67 rho=CORR(z,n,0,n,1,r0,per)
68 PLOTSIZE(7)                ;top half
69 BATCHOFF                   ;do want pause after 2nd plot
70 PLOTSP(per,n,r0)
71 PLOTTOFF                    ;switch back to text mode for listing
72 per=4*per/n
73 LABEL(per)='Periodogram'
74 LISTSP(per,n)              ;list periodogram

```

Example 1.5

TESTING FOR WHITE NOISE

In this example we construct two graphs that are useful for determining whether a data set could be white noise. We will see in Sections 3.1 and 3.2 that if $x(1), \dots, x(n)$ is a random sample from a population, then for large n :

a) The correlations $\hat{\rho}(1), \dots, \hat{\rho}(m)$ are independent and identically distributed as $N(0, 1/n)$ variables. Thus there is approximately a 95 percent chance that an individual $\hat{\rho}(v)$ will be outside of $\pm 1.96/\sqrt{n}$. To produce simultaneous confidence bands having 95 percent confidence level, we must construct individual intervals having level $.95^{1/m}$. This is what is done in the first part of the macro.

b) The cumulative periodogram has a 95 percent chance of falling entirely within the lines $y = 2x \pm 1.36/\sqrt{q}$, where $q = [n/2] + 1$. The second part of the macro constructs the graph of the cumulative periodogram and superimposes these two lines together with the line $y = 2x$.

There are several features to note about **WNTTEST.MAC**. First it uses the **PLOTK** command to do the superimposing of the plots. Second, if the user wants to execute only the second part of the macro (the part starting with the line **;per**), the macro can be invoked with the command **MACRO(wntest,per)**. Third, the macro creates a large number of variables that would be of no use outside of itself. Thus before exiting, **WNTTEST.MAC** deletes these variables using the **CLEAN** command. In Figure 1.14 we give the result of calling the macro with two data sets; the first being a normal white noise series of length 100, and the second being the same series with a cosine of length 100, amplitude .5,

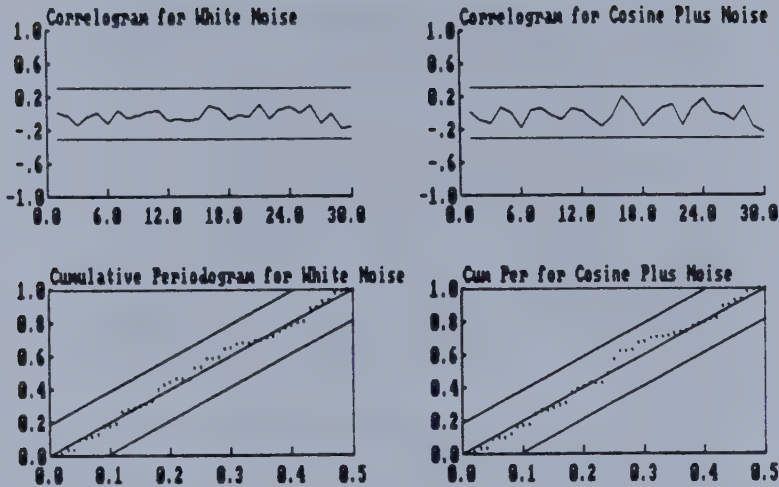


Figure 1.14. Example Output from WNTTEST.MAC.

and period 4 added to it. Notice that none of the boundary lines are crossed for either series, but that the cumulative periodogram seems to increase rapidly at $\omega = .25$, something that is unusual and should be a hint that this data set is in fact not white noise.

```

1 ;;
2 ;;   WNTTEST.MAC: macro to find 95% simultaneous confidence
3 ;;               bands for the correlogram and cumulative
4 ;;               periodogram under the hypothesis of white noise.
5 ;;
6 ;;   Input: x,n,m  (data, sample size, and number of correlations)
7 ;;
8 ;;   Output: rho,R0,per,cper
9 ;;
10 PAUSE
11 ;start
12 rho=CORR(x,n,m,0,1,r0,per)
13 xx=LINE(m,0,1)
14 xx=<xx,1,m,1,m>      ;xx are values on horizontal axis
15 u={1+.95~{1./m}}/2  ;want quantile for u={1+.95~{1/m}}/2
16 zalph=DIST(z,3,1,u)  ;zalph is that quantile
17 lim2=zalph/{n^.5}
18 lim1=-lim2
19 yy=<rho,lim1,lim1,lim2,lim2>
20 nn=<m,2,2>
21 type=<2,2,2>          ;want line plots
22 LABEL(xx)=<x>         ;put label of data into xx

```



```

23 LABEL(yy)='Correlogram and 95% Simultaneous Bands'
24 PLOTK(xx,yy,nn,3,type,0,m,-1,1)
25 CLEAW(u,zalph,lim1,lim2,xx,yy,nn,type)
26 ;
27 ;per
28 ;
29 yy=SUBWWS(x,n,1,xbar)
30 q={n/2}+1
31 FFT(yy,n,q,1,zr,zi)
32 per={zr^2+zi^2}/n
33 CLEAW(yy,zr,zi,xbar)
34 cper=CUMSP(per,n)
35 freq=LINE(q,0,.5,1) ;this gets q frequencies in [0,.5]
36 sq1=1.36/{q^.5}      ;1.36 is the 95% critical value
37 sq2=1-sq1           ;the sq's are some endpoints on graph
38 sq3=.5*sq2
39 sq4=.5*sq1
40 xx=<freq,0,.5,0,sq3,sq4,.5,0,.5,.5,.5>
41 yy=<cper,0,1,sq1,1,0,sq2,1,1,1,0>
42 nn=<q,2,2,2,2,2>
43 type=<34,2,2,2,2,2> ;all plots line plots except cum. per.
44 LABEL(xx)=<x>       ;put label of data into xx
45 LABEL(yy)='Cumulative Periodogram and 95% Bands'
46 PLOTK(xx,yy,nn,6,type)
47 CLEAW(xx,yy,freq,nn,type,q,sq1,sq2,sq3,sq4)

```

Example 1.6 MISSING DATA

The problem of how to treat missing data in time series analysis is very difficult (see Problem T3.18 and Parzen (1983b) for more information about handling the problem). As an example of what can be done with missing data, the macro below replaces any element of a set of data that is zero by the average of the nonzero values. Note that one can enter a decimal point in a TIMESLAB data file for a missing value and TIMESLAB will read it as a zero.

```

1 ;;
2 ;; MISSING.MAC: macro to fill in missing values in a data set
3 ;;           with values that are equal to the mean of the
4 ;;           nonmissing values.
5 ;;
6 ;; INPUT: x, n (data set and sample size)
7 ;;
8 ;; NOTE: a value is considered missing if it is identically zero.
9 ;;
10 ;; OUTPUT: y (same as x but has missing values filled in)
11 ;;
12 PAUSE
13 ;start
14 xi=EXTRACT(x,n,0,ne,ni) ;pull off the indices of nonzero values
15 IF(ni.eq.n,end)         ;go to the end if no missing values

```

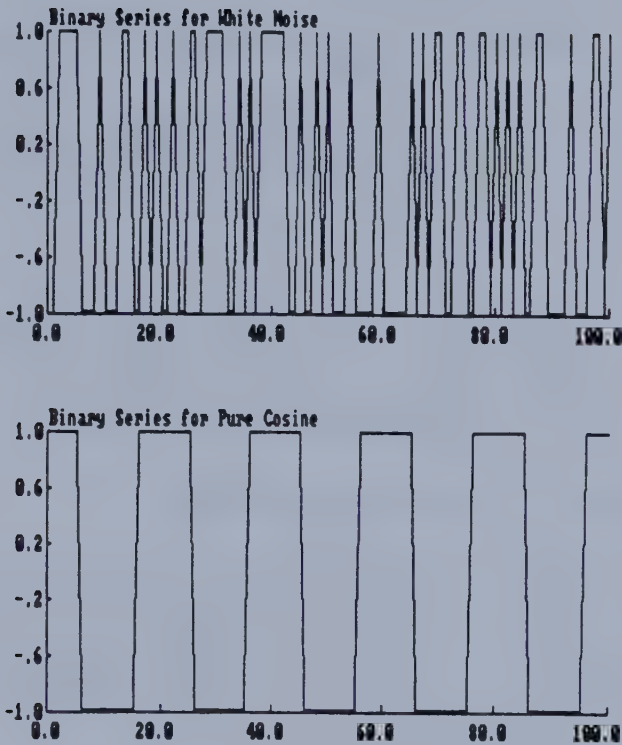


Figure 1.15. The Binary Series Corresponding to a White Noise Series and a Pure Cosine of Period 20.

```

16 xx=EXTRACT(x,xi,ni)           ;now xx is nonmissing values
17 xx=SUBMMS(xx,ni,1,xxbar,0)    ;find average of nonmissing values
18 y=REPLACE(x,n,4,0,0,xxbar)    ;do the replacement
19 ;end

```

Example 1.7 BINARY TIME SERIES

The macro below will take an input time series and produce a binary time series by replacing any value that is less than or equal to a by b and any value that is greater than c by d . If we let $a = c = 0$, $b = -1$, and $d = 1$, then we obtain the plots in Figure 1.15 for a white noise time series of length 100 and a pure cosine of length 100 with period 20. For a description of how one can use such a binary version of a continuous-state time series, see Problem T2.18 and Kedem (1980), p. 34.

```

1 ;;
2 ;;   BINARY.MAC: macro to create a binary version of a data set
3 ;;
4 ;;           x of length n.
5 ;;
6 ;;   INPUT: x, n (original data and sample size)
7 ;;           a, b, c, d (values less than or equal to a are
8 ;;           given the value b, values greater than c are given
9 ;;           the value d)
10 ;;
11 ;;   OUTPUT: y (binary time series)
12 ;;
13 PAUSE
14 ;start
15 xi=EXTRACT(x,n,a,le,n1)    ;xi's are indices of x's less or equal a
16 y=REPLACE(x,xi,b,n1)      ;replace these elements by b
17 xi=EXTRACT(x,n,c,gt,n1)    ;xi's are indices of x's greater than c
18 y=REPLACE(y,xi,d,n1)      ;replace these elements by d
19 LABEL(y)='Binary Series'

```

Example 1.8

THE SAWTOOTH FUNCTION

A classic example of a periodic but nonsinusoidal function is the sawtooth function (see Figure 1.16 for an example). The macro **SAWTOOTH.MAC** will generate a data set consisting of **ncycs** cycles of the sawtooth function where a cycle is defined to be 0 through **amp** followed by **amp-1** through **-amp**, followed by **-amp+1** through 0. Thus each cycle consists of **4*amp** points. The variable **amp** represents the amplitude of the sawtooth. In Figure 1.16, we have used 5 and 10 for **ncycs** and **amp**, respectively.

```

1 ;;
2 ;;   SAWTOOTH.MAC: macro to create a data set containing ncycs
3 ;;
4 ;;           cycles of a sawtooth function having amplitude
5 ;;           amp.
6 ;;
7 ;;   INPUT: ncycs, amp (these must both be integers)
8 ;;
9 ;;   OUTPUT: x, n (data set and sample size, n=4*amp*ncycs)
10 ;;
11 PAUSE
12 ;start
13 ap1=amp+1
14 a2=2*amp
15 x1=LINE(ap1,-1,1)          ;this is 0 thru amp
16 x=x1
17 x1=LINE(a2,amp,-1)         ;this is amp-1 thru -amp
18 x=<x,x1>                    ;append to first part
19 ma=-amp
20 am1=amp-1
21 x1=LINE(am1,ma,1)          ;this is -amp+1 thru -1
22 x=<x,x1>                    ;append

```

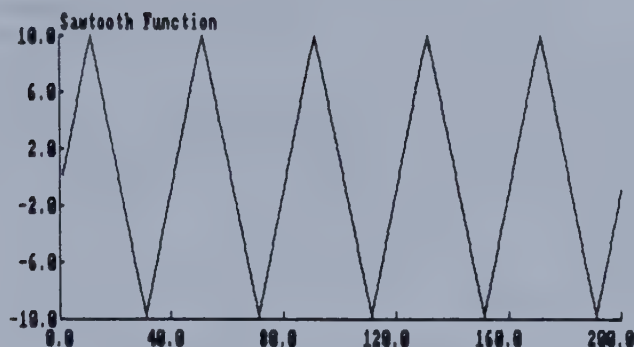


Figure 1.16. An Example of the Sawtooth Function.

```

22 x1=x                ;get ready to form cycles
23 x=<0>
24 nc=1
25 ;
26 ;s1
27 ;
28 x=<x1,x>
29 IF(nc.eq.ncycs,s2)
30 nc=nc+1
31 GOTO(s1)
32 ;
33 ;s2
34 ;
35 n=4*ncycs
36 n=n*amp
37 LABEL(x)='Sawtooth Function'

```

Example 1.9 Two METHODS OF FINDING CORRELATIONS

In Section 1.4 we discussed the convolution and two-FFT methods of calculating sample autocorrelations. The macro below allows a user to experiment with the two methods and obtain precise timings for various sample sizes and numbers of correlations.

```

1 ;;
2 ;;   TIMECOV.MAC: macro to compare the amount of time it takes
3 ;;                   to find the first M+1 sample autocorrelations
4 ;;                   for a data set x of length n by the convolution
5 ;;                   and two-FFT methods.
6 ;;
7 ;;   INPUT: x, n, m, Q (Q must be greater than or equal to n+m and

```

```

8 ;;                                meet the requirements for the use of FFT)
9 ;;
10 PAUSE
11 ;start
12 TIME(t1)
13 rho= CORR(x,n,m,0,1,r0,per)      ;convolution method
14 TIME(t2)
15 tconv=t2-t1
16 TIME(t1)
17 rho= CORR(x,n,m,q,1,r0,per)      ;2 FFT method
18 TIME(t2)
19 tffts=t2-t1
20 LIST(tconv,t2ffts)

```

Example 1.10 RESIDUALS FROM REGRESSION

The macro given below was used to form Figure 1.9.

```

1 ;;
2 ;;   ARTREG.MAC: this macro was used to construct Figure 1.9.
3 ;;
4 ;;   INPUT: none
5 ;;
6 PAUSE
7 ;start
8 READ(artif,y,n) ;read data
9 x1=LINE(n,1,0) ;intercept
10 x2=LINE(n,0,1) ;linear term
11 x3=COS(n,1,10) ;cosine term
12 x=<x1,x2,x3>
13 REG(y,x,n,3,beta,rss,t,e)
14 PLOT(e,n,0,100,-4,4)

```

Example 1.11 MOVING AVERAGE SMOOTHER

The macro SMOOTH.MAC was used to produce the result of using the moving average smoother in Figure 1.10.

```

1 ;;
2 ;;   SMOOTH.MAC: macro to produce a data set consisting
3 ;;               of the sum of a cosine (amplitude 1,
4 ;;               period 40) plus  $N(0,1)$  white noise and
5 ;;               smooth it using a moving average smoother
6 ;;               of 11 points.
7 ;;
8 ;;   INPUT: none
9 ;;
10 PAUSE

```



```

11 ;start
12 y=COS(100,1,40)
13 wn=WN(12345,100)
14 y=y+wn ;y is now cosine plus white noise
15 beta0=1./11.
16 beta=LINE(10,beta0,0)
17 yy=FILT(y,beta,beta0,100,10) ;apply moving average smoother
18 xx=EXTRACT(y,6,95) ;pull off 6th through 95th values
19 yy=<yy,xx> ;prepare the plot
20 xx=LINE(90,5,1)
21 xx=<xx,xx>
22 LABEL(yy)= 'Data (dotted) and Smoothed Data (solid)'
23 LABEL(xx)= ' '
24 type=<2,33>
25 PLOTK(xx,yy,90,2,type,0,100,-3,3) ;do the plot

```

Example 1.12 EXPONENTIAL SMOOTHING

The macro EXPSM.MAC determines the value of α to be used in simple exponential smoothing and uses it to smooth the data.

```

1 ;;
2 ;;   EXPSM.MAC: macro to evaluate the sum of squares of errors
3 ;;           for simple exponential smoothing for k equally
4 ;;           spaced values between a1 and a2 (inclusive)
5 ;;           of the smoothing parameter alpha.
6 ;;
7 ;;           The sum of squares function is plotted and the
8 ;;           minimizing value of alpha used to smooth the data.
9 ;;           A plot of the raw and smoothed data is also given.
10 ;;
11 ;;   INPUT: x, n (data to be smoothed and sample size)
12 ;;           a1, a2, k (first and last values of alpha and how
13 ;;           many values to use)
14 ;;
15 ;;
16 PAUSE
17 ;start
18 PROMPTOFF
19 SS=LINE(k,0,0) ;save space for sum of squares
20 vals=LINE(k,a1,a2,1) ;vals is values to use
21 i=1 ;initialize counter
22 ;
23 ;startloop
24 ;
25 a=vals[i]
26 oa=a-1 ;get ready for difference equation
27 alpha=<oa>
28 w=a*x
29 w=<x[1],w>
30 xhat=ARDT(alpha,1,n,w) ;difference equation

```

```

31 e=x-xhat
32 SS[i]=DOT(e,e,n)           ;sum of squares
33 LIST(a,SS[i])
34 IF(i.eq.k,endlloop)
35 i=i+1
36 GOTO(startloop)
37 ;
38 ;endlloop
39 ;
40 LABEL(SS)='Sum of Squares Function'
41 LABEL(vals)=' '
42 PLOT(vals,SS,k)
43 MAXMIN(SS,k,smax,imax,smin,imin)
44 a=vals[imin]
45 oa=a-1
46 alpha=<oa>
47 w=a*x
48 w=<x[1],w>
49 xhat=ARDT(alpha,1,n,w)
50 LABEL(xhat)=' '
51 PLOT2(x,xhat,n,n,2,1)
52 PROMPT

```

Computational Problems

C1.1. Use the **EDIT** command to form a file containing a time series of length at least 50 from your area of interest and then use the **DESCRIBE** macro to do an initial analysis of the data. Do there appear to be trends or cycles in your data? Do your data appear to be long or short memory? If there are any obvious transformations that might be used on your data, try them and then analyze the transformed data using **DESCRIBE**.

C1.2. In the **WN** command, **TIMESLAB** generates Cauchy variables by $X = \tan(\pi(U - \frac{1}{2}))$ where U is a uniform random variable on the interval $(0,1)$. It can also be shown that the ratio of two independent standard normal variables has the Cauchy distribution. To illustrate this, generate samples of size 200 using both methods and use the **INFQNT** command to produce quantile plots for each sample. Do the plots look the same?

C1.3. An example of a long-memory time series is what is called a random walk process (see Section 2.5.1). A simple example of such a process is to let $x(t)$ be the number of heads minus the number of tails that have occurred after t flips of a coin. Generate a realization of length 200 from such a process by doing the following steps: (1) use the **WN** command to generate 200 observations from a $U(0,1)$ distribution, (2) use the **REPLACE** command with **iopt** = 5 to

convert these to the values $x(t) = -1$ if $U(t) \leq .5$ and $x(t) = 1$ if $U(t) > .5$, and (3) use the **CUM** command to find $S(t) = \sum_{i=1}^t x(i)$ for $t = 1, \dots, 200$. Plot the result. Do you see why this process is called a random walk? Find the correlogram of S and notice how it decays slowly to zero.

C1.4. Use the **PLOT2** command to superimpose plots of a cosine and sine of length 100 with amplitude 1 and period 20. How much out of phase are the two plots? Notice that if you were to add the two curves together, the maximum values in the result would have to be between those in the individual curves. Now add the two curves together and superimpose the plot of all three curves (you will have to use the **PLOTK** command to do this). Refer to Problem T1.3 for more information about this phenomenon.

C1.5. Use the **WN** command to generate a Gaussian white noise series of length 100 and the **CORR** command to find the periodogram of the series. Use the **PLOTSP** command with third argument .001, .01, .1, 1, 10, 100, and 1000, and note how this moves the graph up and down. If we use the sample variance of the process as the third argument, then the curve will be centered around zero on the vertical axis. See Theorem 1.4.3 for more information about this.

C1.6. Use the macro **SAWTOOTH.MAC** (see Example 1.8) to generate an array containing the sawtooth function of length 96 points having amplitude 2. Use the **CORR** and **PLOTSP** commands to produce a plot of the standardized log periodogram of the data. Are there harmonics in the plot? What is the fundamental frequency? Why?

C1.7. Write a macro that will generate a cosine curve \mathbf{x} of length 100 points having period 5 and amplitude 1 and will plot the correlogram of \mathbf{x} for $M=40$. What is $\hat{R}(0)$? Does this agree with what Problem T1.6 says it should be? Note that this problem verifies that the correlogram of a sinusoid is also a sinusoid.

C1.8. Write a macro that will do the following:

- 1) Generate and superimpose the plots of cosines of length n , amplitude 1, and periods p_1 and p_2 for user-specified values p_1 and p_2 of p_1 and p_2 .
- 2) Use the **DOT** command to find the sum of crossproducts SS of the two arrays in part (1).

Use the macro for $n = 100$, and $p_1 = 10$, $p_2 = 20$ and also $p_1 = 10$, $p_2 = 10$. Do the results agree with part (a) of Theorem 1.4.1?

C1.9. Use the **TIMECOV.MAC** macro to find values of **n**, **M**, and **Q** for which the convolution method is faster, and another set for which the two-FFT method is faster.

C1.10. Use the **LINE** command to form an array **x** of length 100 containing **x[i]=i**. Use the **FFT** command to find the discrete Fourier transform of **x** and then find the inverse discrete Fourier transform of the transform of **x**. Verify that the imaginary part of this second transform is zero and the real part is 100 times the original array **x**.

C1.11. Write a macro that will use the **FILT** command to form

$$y(t) = \sum_{j=-k}^k \alpha_j x(t-j), \quad t = 1, \dots, n,$$

where $\alpha_j = \rho^{|j|}$ and ρ , n , k , and x are specified by the user of the macro. Notice that **FILT** is designed to have only coefficients with nonnegative indices and so you will have to be clever in how you form the **beta** and **beta0** to be used by **FILT**. (Hint: **beta0** will be α_{-k} .) Note also that the input array will have to be of length $n + 2k$. Try out the macro for white noise input and for $\rho = .8$ and $\rho = .5$ with $k = 1, 2, 3$ for each ρ .

C1.12. Use the **POLY** command to form a time series consisting of 100 values of a quadratic polynomial over the interval $[0,1]$ (you can choose the coefficients). Use the **DIFF** command twice in order to apply first differencing twice. Is the result a constant?

C1.13. Form a series

$$x(t) = 1 + .01t + \cos\left(\frac{2\pi(t-1)}{10}\right), \quad t = 1, \dots, 100,$$

and use the **EXTEND** command with **d1=1** and **d2=10** to get the next 30 values of the series. Use the **PLOT2** command to superimpose plots of the original data and the extended series.

C1.14. Verify that the following macro gives the same result for **n=100**, **amp=2.0**, and **per=5.0** as does the command

x=COS(n,amp,per).

Note that the macro uses the result of Problem T1.12 to generate a cosine curve.

```

1 ;;
2 ;;   COSDE.MAC: macro to generate a cosine curve of length n having
3 ;;               amplitude amp and period per using the difference
4 ;;               equation method.
5 ;;
6 ;;   INPUT: n, amp, per
7 ;;
8 PAUSE
9 ;start
10 theta=2*pi/per
11 f=line(n,0,0)
12 f[1]=1.
13 f[2]=COS(theta)
14 ct=-2*f[2]
15 alpha=<ct,1>
16 x=ARDT(alpha,2,n,f)
17 x=amp*x

```

C1.15. Use the **WNTTEST.MAC** macro on the first 23 and last 22 years of the Lake Erie levels data (use the **EXTRACT** command to extract the two parts of the series). Does there appear to be any difference in the two analyses?

C1.16. Suppose that X is a random variable having the binomial distribution with parameters n and p ; that is,

$$f(x) = \Pr(X = x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, 1, \dots, n.$$

a) Show that $g(x) = \log f(x)$ satisfies the difference equation

$$g(x) - g(x-1) = h(x) = \log \left(\frac{n-x+1}{x} \frac{p}{1-p} \right), \quad x \geq 1.$$

(Hint: Find $f(x)/f(x-1)$.)

b) Find $g(0)$ and thus use the **ARDT** command to write a macro that will find $g(1), \dots, g(n)$, and thus $f(0), f(1), \dots, f(n)$ for user-specified n and p . Note that you will have to add 1 to all of the indices as arrays in **TIMESLAB** all begin with the index 1.

c) Check the macro by comparing its results with that of using the **BINOM** command for $n=100$ and $p=.4$.

Note that other discrete probability distributions can be found in a similar fashion.

C1.17. Use the **REG** command to find the residuals from a linear trend for the log of the airline data. Does this regression remove the seasonal cycle in the data? From these residuals, construct two data sets: (1) the 12th differences and (2) the result of subtracting monthly means. Use the **WNTTEST.MAC** macro to analyze each of these two series. Which of the two series looks more like white noise?

C1.18. Write a macro that will do the following:

- 1) Generate a data set of length n made up of the sum of a linear trend having intercept **a** and slope **b**, a pure cosine having amplitude **amp** and period **per**, and a white noise series having variance **sig2**.
- 2) Use the **REG** command to find the residuals from regressing the data on a linear trend and a cosine.
- 3) Use the **WNTTEST.MAC** macro to investigate the whiteness of these residuals.

Try out the macro for a variety of coefficients. Find a set of coefficients so that the linear part predominates, another so that the sinusoid does, and another so that the error series does.

C1.19. Extend the Beveridge wheat price data by 120 points using the inverse differencing method with $d = 1$. Plot the results with the horizontal and vertical axes ranging from 0 to 500 and 0 to 500, respectively. Use the **REG** command to fit a linear trend to the data and then extend this line by 120 points and plot the data and the line. Use the same scaling factors as in the inverse differencing method. How do the two plots compare?

C1.20. Write a macro that will calculate and plot the exponential smoothing weights

$$\beta_j = \alpha(1 - \alpha)^{j-1}, \quad j = 1, \dots, n,$$

for user specified values of $\alpha \in (0, 1)$ and n . The easy way to calculate the β 's is to find the logarithms and then use the **EXP** command to get their actual values. For what value of α do the weights decay to zero the slowest?

C1.21. Write a macro that will successively generate white noise data of length 100, 200, 400, 800, 1600, and 3200 and use the **HIST** command to plot the histogram of each series. Let **HIST** choose the number of bins each time. You should be able to do this in an implied loop using the **IF** and **GOTO** commands. Note how the number of intervals chosen by **HIST** increases as the sample size increases. The rule that is used is that the number of intervals is proportional to $n^{1/3}$ where n is the sample size.

C1.22. Generate a normal white noise series of length 200 and use the **DENSITY** command with **rbins=4** and the Epanechnikov kernel to find a probability density estimate at the 51 equally spaced points between -4 and 4 inclusive. Use the **DIST** command to evaluate the true density at the same 51 points, and then use the **PLOTK** command to superimpose plots of the estimated and true densities. Use a point plot (with lower triangles as the symbol at the points) for the density estimate and a line plot for the true density.

C1.23. Generate a pair of independent Gaussian white noise series of length 100. Consider the elements of these series as the real and imaginary parts of a complex time series of length 100. Use the **POLAR** command to find the amplitude (modulus) and phase (the arctangent of the imaginary part divided by the real part) of these complex numbers. Use the **INFQNT** command to find the informative quantile plot of the phases. Does it closely follow the uniform line on the plot? Relate this to Problem T1.4.

C1.24. There has been some controversy as to whether the distribution of the Buffalo snowfall data is unimodal or trimodal. Use the **DENSITY** command with the Gaussian kernel and all combinations of 16, 32, and 48 for **npts** with **rbins=3, 4, and 5** to estimate the density of the data. Do you have an opinion as to how many modes the distribution has? What is the bandwidth of the kernel when **rbins=4**?

C1.25. The matrix quantities discussed in Section A.1 and calculated by the **GS**, **MCHOL**, and **SWEEP** commands are related in a variety of ways. To illustrate this, perform the following experiment.

a) Generate an array **X** of length 40 containing a white noise series. We will think of **X** as the (10×4) matrix **X**, whose first column consists of the first ten elements of the array **X**, whose second column is the next ten elements, and so on. Use **LISTM** to display the matrix **X**.

b) Use **GS** to find the factors **Q** and **R** in the Gram-Schmidt decomposition of **X**. Use the arrays **Q** and **R** to contain the matrices **Q** and **R**. Verify that $\mathbf{X} = \mathbf{QR}$ by using **MMULT** to find the product and **LISTM** to display it. Also check that **Q** is orthogonal by using **MMULT** to find $\mathbf{Q}^T \mathbf{Q}$.

c) Use **MMULT** to find the array $\mathbf{X}^T \mathbf{X}$ representing $\mathbf{X}^T \mathbf{X}$, and **MCHOL** to find the Cholesky factors **L** and **D**. Verify that $\mathbf{L} = \mathbf{R}^T$ and $\mathbf{D} = \mathbf{Q}^T \mathbf{Q}$.

d) Now use **SWEEP** to sweep $\mathbf{X}^T \mathbf{X}$ on its first diagonal and note that the second through fourth elements of the first row of the result are the same as the corresponding elements of **R**, and thus as the below-diagonal elements of the first column of **L**. Next sweep the resulting matrix on its second diagonal and find how the result is related to **R** and **L**. Continue this process until all

of the diagonals have been used. Verify that the result is the inverse of $\mathbf{X}^T \mathbf{X}$ by multiplying it by $\mathbf{X}^T \mathbf{X}$ and displaying the result.

Theoretical Problems

T1.1. Show the following by induction. Also, $\sum_{t=1}^n c = cn$

$$\text{a) } \sum_{t=1}^n t = \frac{n(n+1)}{2}$$

$$\text{b) } \sum_{t=1}^n t^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\text{c) } \sum_{t=1}^n t^3 = \frac{n^2(n+1)^2}{4}$$

$$\text{d) } \sum_{t=1}^n t^4 = \frac{(n+1)(2n+1)(3n^2+3n-1)}{30}$$

Thus find an expression for the correlogram of $x(t) = a + bt$, $t = 1, \dots, n$.

T1.2. Let $f(\theta) = a \cos \theta + b \sin \theta$, and let $c = \sqrt{a^2 + b^2}$. Define the $\arctan 2$ function of a and b by:

$$\phi = \arctan 2(b, a) = \begin{cases} \arctan(b/a), & a > 0 \\ \arctan(b/a) + \operatorname{sgn}(b)\pi, & a < 0 \\ \operatorname{sgn}(b)\frac{\pi}{2}, & a = 0, \end{cases}$$

where $\operatorname{sgn}(b)$ is 1 for $b \geq 0$ and -1 for $b < 0$, while \arctan is the usual inverse tangent function which takes on values only in the interval $(-\pi/2, \pi/2)$.

Show that for all a and b ,

$$a = c \cos \phi, \quad b = c \sin \phi,$$

and thus

$$f(\theta) = c \cos(\theta - \phi).$$

(Hint: $\cos(\theta_1 - \theta_2) = \cos \theta_1 \cos \theta_2 + \sin \theta_1 \sin \theta_2$.) Throughout this book, when we write \arctan we mean $\arctan 2$.

T1.3. Show that the sinusoid

$$f(x) = a \cos \frac{2\pi x}{p} + b \sin \frac{2\pi x}{p}$$

has maximum values at $x = (p\phi/2\pi) + kp$, for any integer k , where $\phi = \arctan 2(b/a)$. Show that if $a > 0$, then $\phi/2\pi$ (or $1 - \phi/2\pi$ if $b < 0$) is the fraction of a cycle that f is shifted to the right of $a \cos 2\pi x/p$. Thus show that if $a > 0$, then f can only be shifted to the right of $a \cos 2\pi x/p$ a fraction of a cycle contained in either the interval $(0, 1/4)$ or the interval $(3/4, 1)$. Show that the same thing is true for $a < 0$.

T1.4. Show that if a and b are independent random variables each having the $N(0, 1)$ distribution, then $c^2 = a^2 + b^2$ and $\theta = \arctan(b/a)$ are independent random variables with c^2 being χ_2^2 and θ being $U(-\pi, \pi)$.

T1.5. Show that $e^{i\theta} = \cos \theta + i \sin \theta$, and thus that $e^{2\pi i k} = 1$ for any integer k .

T1.6. Let $x(t) = a \cos 2\pi(t-1)/p$, $t = 1, \dots, n$, where $n = mp$; that is, the x 's consist of m cycles of a pure cosine with amplitude a and integer period $p > 2$.

a) Show that $\bar{x} = 0$.

b) Show that for $0 \leq v < n$,

$$\hat{R}(v) = a^2 \frac{1}{2} \left(1 - \frac{v}{n}\right) \cos \frac{2\pi v}{p} - \frac{c}{2n} \sin \frac{2\pi v}{p},$$

where $c = \frac{1}{2 \sin \frac{2\pi}{p}}$, and thus show that

$$\hat{\rho}(v) = \left(1 - \frac{v}{n}\right) \cos \frac{2\pi v}{p} - \frac{c}{n} \sin \frac{2\pi v}{p}.$$

Thus note that the correlogram of a cosine is itself a damped cosine having the same period (damped because of the factor $1 - (v/n)$) minus a sine having amplitude going to zero as n gets large.

(Hints: If z is a complex number, then $1/z = \bar{z}/|z|^2$. Use real part and geometric series arguments as in Theorem 1.4.1. Note that $\cos 2\theta = \cos^2 \theta - \sin^2 \theta$.)

T1.7. Suppose that

$$y_t = \mu_{j(t)} + \epsilon_t, \quad t = 1, \dots, n = kd,$$

where the ϵ 's are iid $N(0, \sigma^2)$ and $j(t) = \text{mod}(t-1, d) + 1$, where

$$\text{mod}(l, m) = l - m[l/m],$$

and $[x]$ denotes the greatest integer less than or equal to x .

a) Write this model as $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{\beta} = (\mu_1, \dots, \mu_d)^T$ and \mathbf{X} is $(n \times d)$.

b) Show that the least squares estimates $\hat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ are the seasonal means described in Section 1.5, and that $\hat{\boldsymbol{\beta}} \sim N_d(\boldsymbol{\beta}, \sigma^2 \mathbf{I}_d)$.

c) Suppose $d = 7$ and

$$y_t = \mu + \sum_{k=1}^3 \left[a_k \cos \frac{2\pi(t-1)k}{7} + b_k \sin \frac{2\pi(t-1)k}{7} \right] + \epsilon_t,$$

for $t = 1, \dots, n = 7m$. Let $\boldsymbol{\gamma} = (\mu, a_1, a_2, a_3, b_1, b_2, b_3)^T$. Write this model as $\mathbf{y} = \mathbf{W}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$, find the least squares estimator of $\boldsymbol{\gamma}$, and thus show that

$$\frac{n}{4}(\hat{a}_k^2 + \hat{b}_k^2) = \hat{f}(\omega_{km+1}),$$

where \hat{f} denotes the periodogram of the y 's.

T1.8. Show that if we calculate the IDFT of the DFT of a set of n numbers, then the result is n times the original set of numbers.

T1.9. Show that the s th difference of a set of data that is periodic with period s is no longer periodic.

T1.10. Show that in the exponential smoothing model,

$$\hat{x}(t+1) = \alpha x(t) + (1-\alpha)\hat{x}(t).$$

T1.11. Show that if the zeros of the characteristic polynomial for a homogeneous difference equation for z are all greater than one in modulus, then $\lim_{t \rightarrow \infty} z(t) = 0$.

T1.12. Use Theorem 1.6.1 to show that the difference equation

$$z(t) - 2 \cos \omega z(t-1) + z(t-2), \quad t \geq 2,$$

with $z(0) = 1$ and $z(1) = \cos \omega$ has solution $z(t) = \cos t\omega$.

T1.13. Show that the zeros of the polynomials

$$g(z) = \sum_{j=0}^p \alpha_j z^j \quad \text{and} \quad h(z) = \sum_{j=0}^p \alpha_j z^{p-j}$$

are the reciprocals of each other.

T1.14. If we form m independent confidence intervals, each having confidence coefficient $1 - \alpha$, what is the probability that all m of the intervals will include the value of the parameter that they are the confidence interval for? Thus, what should α be in order to have 20 confidence intervals have probability .95 of including the true value of their parameters?

T1.15. The informative quantile function of a continuous random variable X (as opposed to that for a data set) is defined by

$$IQ(u) = \frac{Q(u) - Q(.5)}{2(Q(.75) - Q(.25))}, \quad u \in (0, 1),$$

where Q is the quantile function of X . The quantile function is defined by, assuming the cdf F of X is strictly increasing,

$$Q(u) = x \quad \Longleftrightarrow \quad F(x) = \Pr(X \leq x) = u.$$

Show that the true informative quantile plot of any uniformly distributed random variable is the same. Show that the reference line placed on the plot produced by **INFQNT** is this informative quantile plot.

T1.16. Show that the Parzen kernel is the probability density function of the average of four independent random variables, each having the uniform distribution on the interval $[-1, 1]$. (Hint: Find the moment generating function of the average.)

CHAPTER 2

Probability Theory for Univariate Time Series

In this chapter we consider how the basic ideas of probability theory apply to the analysis of time series. In Section 2.2 we give the basic definition of covariance stationarity and introduce the theoretical autocorrelation and spectral density functions. In Sections 2.3 and 2.4 we discuss linear filters and prediction theory, and in Section 2.5 we discuss in detail the models that are traditionally used for data. Finally, in Section 2.6 we describe a variety of algorithms for univariate time series.

2.1. Introduction

As in many areas of statistics, our basic aim in time series analysis is twofold: descriptive and inferential. In Chapter 1 we considered some basic ways to describe time series data. To make inferences from data we will use the following strategy:

1. Assume that some member of a family of models will adequately represent the observed behavior of a time series data set.
2. Identify which member of the family best represents the data (model identification).
3. Estimate the parameters of the chosen model.
4. Check the adequacy of the fit of the estimated model.
5. Make statistical and scientific inferences based on the characteristics of the chosen model.

Note that we will also make inferences occasionally without assuming a particular model for the data. We will call these nonparametric in analogy

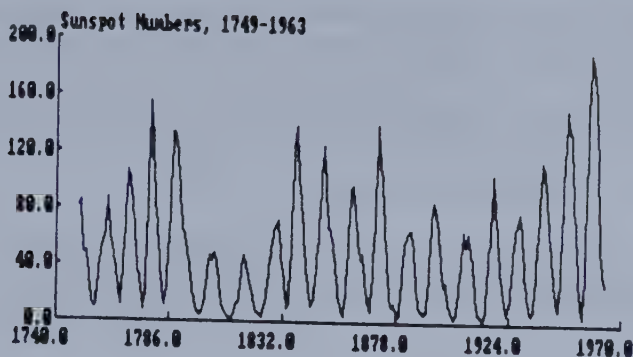


Figure 2.1. Sunspot Numbers, 1749-1963.

with the usual nonparametric analysis in the random sampling case.

The aim of this chapter is to introduce both the quantities that we want to make inferences about (the correlogram, spectral density function, and predictors) and the models that are usually used to allow us to make meaningful inferences about these quantities.

In the typical random sampling model of elementary statistics, one's aim is to make inferences about population parameters from statistics calculated from random samples. In time series analysis the ideas of population and sample are not so clear cut. We visualize that a data set $x(1), \dots, x(n)$ is just one possible set out of many that could have been generated by some random mechanism that is producing data. The set of all possible realizations that could be observed is called the ensemble of realizations. Figure 2.1 is a time plot of a set of sunspot numbers observed annually from 1749 to 1963. We assume that annual total sunspot numbers are a random function of time—that is, that the observation in any one year is a random variable. Our aim is to make inferences about the joint behavior of these random variables when we have only one observation of each one. We can do this by developing ideal models for their joint distributions. In elementary statistics we usually need not consider joint behavior of observations because they are assumed independent.

Definition. A time series is an indexed collection $\{X(t), t \in T\}$ of random variables having finite second moments; that is, $E(X^2(t)) < \infty$ for each element of the index set T .

Usually we will consider T to be the set \mathcal{Z} of all integers; that is, we will assume that the phenomenon being observed has been going on for a long time and will continue indefinitely. We will often refer to a time series as

$X(t)$ or just X if there is no possibility of confusion. We use capital letters to refer to random variables and lowercase letters for particular values for the random variables. For the sunspot data, $X(1)$ refers to the random variable representing possible sunspots in 1749, while $x(1)=80.9$ denotes the actual observed value in 1749.

2.2. Covariance Stationary Time Series

Traditionally, two assumptions are made about the joint behavior of the random variables making up a time series. The first we will label as an assumption and the second is the definition of covariance stationarity.

Assumption. *The behavior of a time series X can be adequately described by a knowledge of its mean value function m and covariance kernel K , that is,*

$$\begin{aligned} m(t) &= E(X(t)), & t \in \mathcal{Z} \\ K(s, t) &= \text{Cov}(X(s), X(t)), & s, t \in \mathcal{Z}. \end{aligned}$$

If X is a Gaussian time series then this assumption is valid. We will see later that a wide variety of non-Gaussian time series can also be effectively analyzed. There is currently a great deal of effort being made in the time series community toward developing methods for analyzing data from processes which do not satisfy this assumption (see Priestley (1981, pp. 866-890), for example).

Definition. *A time series X is said to be a Gaussian time series if the joint distribution of any finite number of $X(t)$'s is multivariate normal; that is, for any positive integer n and any n integers t_1, \dots, t_n , we have that the joint distribution of $X(t_1), \dots, X(t_n)$ is multivariate normal.*

The multivariate normal distribution plays a crucial role in the theory and analysis of time series. A discussion of its basic properties is given in Section A.4.

Definition. *A time series X is said to be covariance (or weakly) stationary if its mean value function is a constant, that is, $E(X(t)) = \mu$, and if there exists a function $\{R(v), v \in \mathcal{Z}\}$ such that*

$$K(s, t) = \text{Cov}(X(s), X(t)) = R(t - s);$$

that is, the covariance of any pair of X 's that are the same distance apart is the same. The function R is called the autocovariance function of X .

We will often drop the prefix *auto* in the word *autocovariance*. Note that there is another (stronger) type of stationarity, namely strict stationarity (see Problem T2.1).

The Autocorrelation Function

We usually concentrate on the correlation between two random variables rather than on their covariance.

Definition. Let X be a covariance stationary time series having autocovariance function R . The autocorrelation function of X is given by

$$\rho(v) = \text{Corr}(X(t), X(t+v)), \quad v \in \mathcal{Z}.$$

A plot of $\rho(v)$ versus v for $v = 0, \dots, M$ is called the correlogram of X .

Note that by the definition of the correlation of two random variables, we have

$$\rho(v) = \frac{\text{Cov}(X(t), X(t+v))}{\sqrt{\text{Var}(X(t))\text{Var}(X(t+v))}} = \frac{R(v)}{\sqrt{R(0)R(0)}} = \frac{R(v)}{R(0)},$$

and thus $\rho(0) = 1$ and $\rho(-v) = \rho(v)$. Thus the autocorrelation function of X is just the autocovariance function divided by the variance $R(0)$ of the series.

Toeplitz Matrices

If we have a finite realization $\mathbf{X}_n = (X(1), \dots, X(n))^T$ from a covariance stationary time series X having autocovariance function R , then the covariance matrix of \mathbf{X}_n has the special form

$$\text{Var}(\mathbf{X}_n) = \begin{bmatrix} R(0) & R(1) & R(2) & \cdots & \cdots & R(n-1) \\ R(1) & R(0) & R(1) & \ddots & & \vdots \\ R(2) & R(1) & R(0) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & R(2) \\ \vdots & & \ddots & \ddots & \ddots & R(1) \\ R(n-1) & \cdots & \cdots & R(2) & R(1) & R(0) \end{bmatrix},$$

where the diagonal rows of dots indicate that each element of a particular diagonal is the same; that is, the (j, k) th element of the matrix is $R(|j - k|)$. Such matrices are said to be symmetric Toeplitz and they have been extensively studied (see Grenander and Szegö (1958) and Example 2.1 for example). An $(n \times n)$ symmetric Toeplitz matrix has only n distinct elements, namely the elements $R(0), \dots, R(n - 1)$ that are in its first row. We will use the notation

$$\Gamma_n = \text{Toepl}(R(0), \dots, R(n - 1))$$

for such a matrix. Sometimes we will consider the correlation matrix of \mathbf{X}_n , that is,

$$\Phi_n = \text{Toepl}(1, \rho(1), \dots, \rho(n - 1)).$$

The TOEPL Command

The command

G=TOEPL(R,R0,n)

will form the $(n \times n)$ Toeplitz matrix G with the first row given by the scalar $R0$ followed by the first $n-1$ elements of the array R . Note that the $(n \times n)$ identity matrix is a Toeplitz matrix having first row $(1, 0, \dots, 0)$ and can thus be formed by

```
zero=LINE(n,0,0)
I=TOEPL(zero,1,n)
```

We will be primarily concerned with covariance stationary time series and thus will be trying to make inferences about μ and the autocovariance function R . In Theorem 2.2.1 we summarize some of the basic facts about R , and introduce two functions that are mathematically equivalent to R and are of importance in their own right, both theoretically and in practice. Some of the mathematics in this theorem is quite advanced and is summarized in Section A.2. The interested reader can also consult Bloomfield (1976). Following the theorem is a discussion of its implications.

Theorem 2.2.1 TWO SPECTRAL REPRESENTATIONS

Let $\{R(v), v \in \mathcal{Z}\}$ be the autocovariance function of the covariance stationary time series X . Then

a) $R(v) = R(-v), \quad v \in \mathcal{Z}.$

b) R is a positive semidefinite sequence; that is, for any positive integer n , any real numbers a_1, \dots, a_n , not all zero, and any n time points t_1, \dots, t_n ,

we have

$$S = \sum_{j=1}^n \sum_{k=1}^n a_j a_k R(t_j - t_k) \geq 0.$$

c) SPECTRAL REPRESENTATION OF R . There exists a function $F(\omega)$, $\omega \in [0, 1]$, called the spectral distribution function of X , such that

$$R(v) = \int_0^1 \cos 2\pi v \omega dF(\omega), \quad v \in \mathcal{Z},$$

where the integral is a Lebesgue-Stieltjes integral (see Section A.2.2). Further, $F(0) = 0$, $F(1) = R(0)$, and at all continuity points of F , $F(\omega) = R(0) - F(1 - \omega)$.

d) SPECTRAL REPRESENTATION OF X . There exist two uncorrelated stochastic processes $C = \{C(\omega), \omega \in [0, 1]\}$ and $S = \{S(\omega), \omega \in [0, 1]\}$ having stationary uncorrelated increments (see Section A.2.3) such that $X(t)$ can be written as the stochastic integral

$$X(t) = \int_0^1 \cos 2\pi t \omega dC(\omega) + \int_0^1 \sin 2\pi t \omega dS(\omega),$$

and for $0 \leq \omega_1 \leq \omega_2 \leq 1$,

$$\text{Var}(C(\omega_2) - C(\omega_1)) = \text{Var}(S(\omega_2) - S(\omega_1)) = F(\omega_2) - F(\omega_1).$$

e) If R is absolutely summable, that is, $\sum_{v=-\infty}^{\infty} |R(v)| < \infty$, then F is absolutely continuous; that is, there exists a function $f(\omega)$, $\omega \in [0, 1]$, symmetric about $\omega = .5$, called the spectral density function of X , such that

$$F(\omega) = \int_0^{\omega} f(x) dx,$$

$$f(\omega) = \frac{dF(\omega)}{d\omega},$$

and

$$R(v) = \int_0^1 f(\omega) \cos 2\pi v \omega d\omega, \quad v \in \mathcal{Z}.$$

Further, f is continuous and

$$f(\omega) = \sum_{v=-\infty}^{\infty} R(v) \cos 2\pi v \omega, \quad \omega \in [0, 1].$$

f) The equations relating R and f can also be written as

$$\begin{aligned} R(v) &= \int_0^1 f(\omega) e^{2\pi i v \omega} d\omega \\ f(\omega) &= \sum_{v=-\infty}^{\infty} R(v) e^{-2\pi i v \omega} \\ &= R(0) + 2 \sum_{v=1}^{\infty} R(v) \cos 2\pi v \omega. \end{aligned}$$

g) If $f(\omega) \geq \lambda > 0$ for each $\omega \in [0, 1]$, then R is positive definite; that is, the \geq in part (b) can be replaced by strict inequality.

Partial Proof: Much of the theory leading to the results of this theorem is beyond the scope of this book, particularly parts (c) and (d). The interested reader can consult Priestley (1981, Chapter 4) or Fuller (1976, Chapter 4) for details. Part (a) follows from the fact that $R(v) = \text{Cov}(X(t), X(t+v)) = \text{Cov}(X(t+v), X(t)) = R(-v)$, while part (b) is true because (see Theorem A.4.1)

$$S = \text{Var}(\mathbf{a}^T \mathbf{X}) \geq 0,$$

where $\mathbf{a}^T = (a_1, \dots, a_n)$ and $\mathbf{X}^T = (X(t_1), \dots, X(t_n))$. To see part (g), note that

$$\begin{aligned} S &= \sum_{j=1}^n \sum_{k=1}^n t_j t_k \int_0^1 f(\omega) e^{2\pi i (j-k)\omega} d\omega \\ &\geq \lambda \sum_{j=1}^n \sum_{k=1}^n t_j t_k \int_0^1 e^{2\pi i (j-k)\omega} d\omega. \end{aligned}$$

But this integral is zero unless $j = k$, in which case it is one. Thus

$$S \geq \lambda \sum_{j=1}^n t_j^2 > 0,$$

since we have assumed that not all of the t_j 's are zero.

The alternate expressions for $R(v)$ and $f(\omega)$ in part (f) follow from the facts that $R(v) = R(-v)$ and $\cos(-\theta) = \cos \theta$, $\sin(-\theta) = -\sin \theta$; for example,

$$\sum_{v=-\infty}^{\infty} R(v) e^{-2\pi i v \omega} = \sum_{v=-\infty}^{\infty} R(v) \cos 2\pi v \omega - i \sum_{v=-\infty}^{\infty} R(v) \sin 2\pi v \omega,$$

while

$$\begin{aligned}
 \sum_{v=-\infty}^{\infty} R(v) \sin 2\pi v\omega &= R(0) \sin(0) + \sum_{v=1}^{\infty} [R(v) \sin 2\pi v\omega + R(-v) \sin(-2\pi v\omega)] \\
 &= 0 + \sum_{v=1}^{\infty} R(v) [\sin 2\pi v\omega + \sin(-2\pi v\omega)] \\
 &= 0.
 \end{aligned}$$

The fact that f is symmetric about $\omega = .5$ follows from the fact that $\cos 2\pi v\omega = \cos 2\pi v(1 - \omega)$.

Implications: We are assuming that all inferences about a covariance stationary time series X can be based on making inferences about μ and R . Parts (c)–(e) of the theorem provide us with a function (F or f) that is mathematically equivalent to R . These functions are important in their own right in many circumstances. Note that part (c) provides a harmonic analysis of the infinite sequence R ; that is, R can be represented as a sum (actually a special kind of integral) of sinusoids whose coefficients are “increments” of the spectral distribution function F . Part (d) says that one possible realization $\{x(t), t \in \mathcal{Z}\}$ can be thought of as a sum of infinitely many sinusoids (it is always useful to think of any kind of integral as a sum of a large number of terms) where within a particular realization the amplitudes of these sinusoids are fixed, but between possible realizations they vary according to a probability law. These sinusoids are called frequency components and we can think of $f(\omega)$ as being proportional to the average value (over many realizations) of the squared amplitudes of the frequency component of frequency ω .

We note that F and f are analogous to a cumulative distribution function (cdf) and probability density function (pdf) in probability theory except that since f integrates to $R(0)$, they provide a distribution of variance, not probability. To take this analogy one step further, note that R is in the form of a characteristic function (it is a sequence since f is symmetric on a finite interval).

We will sometimes extend f to the whole real line by making it a periodic function with period 1; that is, $f(\omega + k) = f(\omega)$ for all integers k . It should be noted that much of the literature of time series analysis defines f to be a function on $-\pi$ to π or $-.5$ to $.5$. By doing a change of variables in all of the formulas above, we could obtain the corresponding formulas on other intervals (see Problem T2.8).

Ensemble Mean Interpretation of ρ and f

Perhaps the most useful interpretation of ρ and f is as ensemble averages of the sample autocorrelation and sample spectral density functions introduced in Chapter 1. We will show in Chapter 3 that under very general conditions,

$$E(\hat{f}(\omega)) \rightarrow f(\omega) \quad \text{and} \quad E(\hat{\rho}(v)) \rightarrow \rho(v)$$

as the length of the sample realization goes to ∞ . For example, if we are interested in studying the EEG of a patient, we would recognize that a sample time series consisting of an EEG record would vary from one time to another in that patient's history and we would be most interested in making inferences about some average behavior of the observed time series. In a situation like this, viewing the spectral density function as the average of the sample spectral density over many long realizations of EEG records is a natural thing to do.

To further illustrate this interpretation, consider Figure 2.2, which contains the graph of the true spectral density function f of a time series X , together with three realizations of length 200 from X and the sample spectral density for each of these realizations. Finally, we have included the average of ten such sample spectral densities. Note how each of the sample spectral densities is extremely variable, but the average spectrum follows the true spectral density fairly closely. See Example 2.2 for a macro that further develops the interpretation of the spectral density as the ensemble average of the periodogram.

The White Noise Process

The simplest type of time series model is when X consists of uncorrelated random variables having a mean of 0 and a constant variance σ^2 .

Definition. A time series X is said to be a white noise process with variance σ^2 if

$$E(X(t)) = 0, \quad t \in \mathbb{Z}$$

$$R(v) = \text{Cov}(X(t), X(t+v)) = \begin{cases} \sigma^2, & v = 0 \\ 0, & v \neq 0. \end{cases}$$

Such a process is denoted by $X \sim \text{WN}(\sigma^2)$.

If we define the Kronecker delta function

$$\delta_v = \begin{cases} 1, & v = 0 \\ 0, & v \neq 0, \end{cases}$$

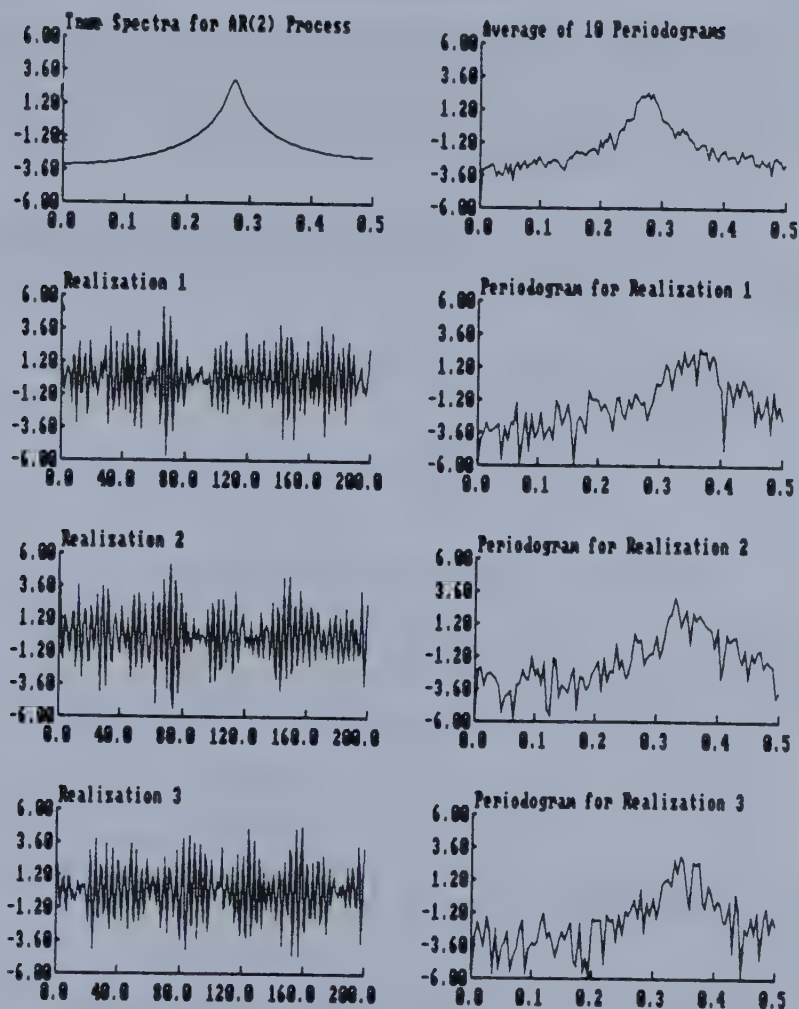


Figure 2.2. Illustrating the Interpretation of the Spectral Density Function.

we can write $R(v) = \delta_v \sigma^2$ when $X \sim \text{WN}(\sigma^2)$. Note that

$$f(\omega) = \sum_{v=-\infty}^{\infty} R(v) \cos 2\pi v \omega = \sigma^2 \sum_{v=-\infty}^{\infty} \delta_v \cos 2\pi v \omega = \sigma^2;$$

that is, the spectral density of a white noise process is flat (a constant). Thus in analogy with the physical spectrum of white light, a sequence of uncorrelated

random variables (which is often used to model physical noise) is referred to as white noise. Processes not having constant spectral densities are often called colored noise.

2.3. The Theory of Linear Filters

We will often express one time series Y as a linear function of the values of another series X . We would like to be able to derive the probabilistic properties of Y from known properties of X .

Definition. The time series Y is a filtered version of the time series X with filter coefficients $\{\beta_j, j \in \mathcal{Z}\}$ if we can write (as a limit in mean square, see Section A.4.4)

$$Y(t) = \sum_{j=-\infty}^{\infty} \beta_j X(t-j), \quad t \in \mathcal{Z}.$$

In Chapter 1 we defined the moving average smoother

$$Y(t) = \sum_{j=-M}^M \frac{1}{2M+1} X(t-j),$$

which is a filtering operation with coefficients

$$\beta_j = \begin{cases} \frac{1}{2M+1}, & |j| \leq M \\ 0, & |j| > M. \end{cases}$$

Another simple example of filtering a time series is a moving average process (as opposed to the moving average smoother).

Definition. The time series Y is called a moving average process of order q if

$$Y(t) = \sum_{k=0}^q \beta_k \epsilon(t-k), \quad t \in \mathcal{Z},$$

where $\beta_0 = 1$ and $\epsilon \sim \text{WN}(\sigma^2)$. We will write $Y \sim \text{MA}(q, \beta, \sigma^2)$ to denote such a series. We will also often write $Y \sim \text{MA}(q)$ to mean that Y is a moving average process of order q without concern for its coefficients β or noise variance σ^2 .

We have seen that if $\epsilon \sim \text{WN}(\sigma^2)$, then $R_\epsilon(v) = \delta_v \sigma^2$ and $f_\epsilon(\omega) = \sigma^2$ where we have now put the subscript ϵ on R and f to indicate to which time

series they correspond. The following theorem will allow us to easily find R_Y and f_Y for an MA process. In fact, the theorem provides expressions for R_Y and f_Y (as long as they exist) in terms of R_X and f_X whenever Y is a filtered version of X . We can further simplify things by introducing what is called the covariance generating function of the autocovariance function R .

Definition. The covariance generating function Γ of a covariance stationary time series X having autocovariance function R is the complex valued function

$$\Gamma(z) = \sum_{v=-\infty}^{\infty} R(v)z^v, \quad z \in \mathcal{C}.$$

Note that $f(\omega) = \Gamma(e^{-2\pi i\omega})$.

Theorem 2.3.1

UNIVARIATE FILTER THEOREM

Suppose that X is a covariance stationary time series with autocovariance function R_X and spectral density function f_X . Suppose that Y is a filtered version of X with filter coefficients β . Then assuming that the quantities involved exist, we have

- a) Y is also covariance stationary.
- b) The autocovariance function of Y is given by

$$R_Y(v) = \sum_{k=-\infty}^{\infty} R_{\beta}(k)R_X(v-k), \quad v \in \mathcal{Z},$$

where

$$R_{\beta}(k) = \sum_{j=-\infty}^{\infty} \beta_j \beta_{j+|k|}, \quad k \in \mathcal{Z}.$$

- c) The covariance generating function of Y is given by

$$\Gamma_Y(z) = \Gamma_X(z)\Gamma_{\beta}(z)\Gamma_{\beta}(z^{-1}),$$

where Γ_X is the covariance generating function of X and

$$\Gamma_{\beta}(z) = \sum_{j=-\infty}^{\infty} \beta_j z^j.$$

d) The spectral density function of Y is given by

$$f_Y(\omega) = |h(e^{2\pi i\omega})|^2 f_X(\omega),$$

where the function

$$h(z) = \sum_{k=-\infty}^{\infty} \beta_k z^k, \quad z \in \mathcal{C},$$

is called the impulse response function of the filter.

Note that part (d) follows immediately from part (c) by substituting $e^{2\pi i\omega}$ for z . We note that $h(e^{2\pi i\omega})$ is called the frequency transfer function of the filter. We will use the Filter Theorem extensively in the sequel, particularly in the next section where we introduce time series models having a finite number of parameters.

R and f for an $\text{MA}(q)$

Let $Y \sim \text{MA}(q, \beta, \sigma^2)$. Then part (a) of the Filter Theorem allows us to immediately conclude that Y is covariance stationary since it is a filtered version of white noise, which is certainly itself covariance stationary. Further we have

$$f_Y(\omega) = \sigma^2 |h(e^{2\pi i\omega})|^2, \quad \omega \in [0, 1],$$

where $h(z) = \sum_{k=0}^q \beta_k z^k$ is a q th-degree complex valued polynomial. From Theorem 2.2.1 we also have

$$f_Y(\omega) = R_Y(0) + \sum_{v=1}^q R_Y(v) \cos 2\pi v\omega.$$

To obtain R_Y , note that

$$R_\beta(v) = \begin{cases} \sum_{j=0}^{q-|v|} \beta_j \beta_{j+|v|}, & |v| \leq q \\ 0, & |v| > q, \end{cases}$$

and thus since $R_\epsilon(v) = \delta_v \sigma^2$, we have

$$\begin{aligned} R_Y(v) &= \sum_{k=-\infty}^{\infty} R_\beta(k) \delta_{v-k} \sigma^2 \\ &= R_\beta(v) \sigma^2 \\ &= \begin{cases} \sigma^2 \sum_{j=0}^{q-|v|} \beta_j \beta_{j+|v|}, & |v| \leq q \\ 0, & |v| > q. \end{cases} \end{aligned}$$

This is an important characterization of an MA(q) process, namely $R_Y(v) = 0$ for $|v| > q$. Note that the covariance generating function of an MA process is $\Gamma_Y(z) = \sigma^2 \Gamma_\beta(z) \Gamma_\beta(z^{-1})$ where Γ_β is a q th-degree polynomial.

The Effect of Differencing

The Filter Theorem is also important for studying the effects of some transformations. For example, we can see the effect of differencing very clearly. Thus suppose X is a covariance stationary time series with spectral density function f_X . Suppose Y is obtained as the d th difference of X ; that is, $Y(t) = X(t) - X(t-d)$. Then Y is covariance stationary and

$$f_Y(\omega) = |1 - e^{2\pi i d \omega}|^2 f_X(\omega),$$

since $h(z) = 1 - z^d$. Thus $f_Y(\omega)$ will be zero anywhere that $e^{2\pi i d \omega}$ is one, namely at any ω such that $d\omega$ is an integer. Thus

$$f_Y\left(\frac{j}{d}\right) = 0, \quad j = 0, 1, \dots, d.$$

In particular, first differencing makes $f_Y(0) = f_Y(1) = 0$, while 12th differencing makes $f_Y(0) = f_Y(\frac{1}{12}) = \dots = f_Y(1) = 0$. Thus differencing totally removes frequency components of these frequencies from a time series. In fact, any differencing makes $f_Y(0) = 0$. We will see in Chapter 3 that the spectral density at zero frequency is an important quantity in making inferences about the mean of a time series. Because of this zeroing out of frequencies, many analysts suggest that great care be taken when using differencing.

What Does the MA Smoother Do?

Another example of using the Filter Theorem to study the effect of transformations is to consider the moving average smoother. Let X be a covariance stationary time series with spectral density f_X and let

$$Y(t) = \sum_{j=-M}^M \frac{1}{2M+1} X(t-j).$$

The frequency response function of this filter is

$$h(e^{2\pi i \omega}) = \frac{1}{2M+1} \sum_{j=-M}^M e^{2\pi i j \omega} = \frac{1}{2M+1} D_M(\omega),$$

where the function

$$D_M(\omega) = \sum_{j=-M}^M e^{2\pi i j \omega}$$

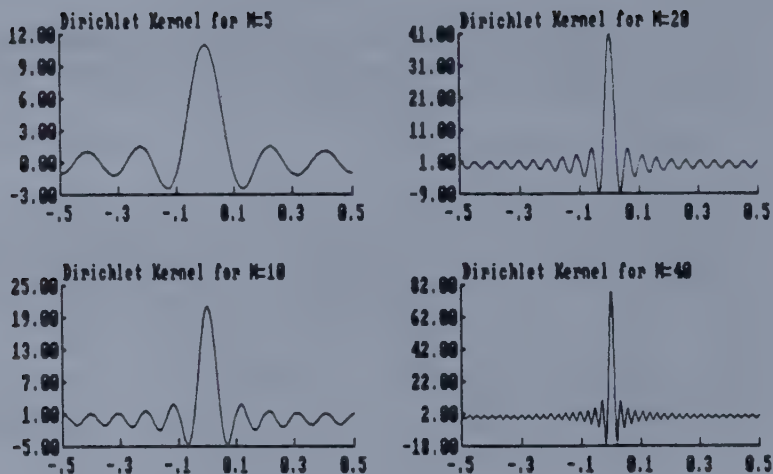


Figure 2.3. The Dirichlet Kernel.

is called the Dirichlet kernel and is well known in many scientific areas. We have (see Problem T2.11)

$$D_M(\omega) = \frac{\sin[(M + \frac{1}{2})2\pi\omega]}{\sin \pi\omega}.$$

Figure 2.3 (see Example 2.3) gives graphs of D_M for $M = 5, 10, 20$, and 40 . Note that the kernel becomes more concentrated about zero as M increases. Note also that the kernel is negative for certain frequency ranges and has large "sidelobes," that is, secondary peaks. Thus we have

$$f_Y(\omega) = \left(\frac{1}{2M+1} \right)^2 |D_M(\omega)|^2 f_X(\omega),$$

and D_M becomes more and more concentrated around frequency zero as M (the number of terms on each side of $X(t)$ used in the average) gets large. Thus the moving average smoother is essentially allowing only frequency components of low frequency to be "passed" to Y from X .

Definition. If Y is a filtered version of X with frequency transfer function $h(e^{2\pi i\omega})$, then the filter is called:

a) a low (high) pass filter if only low (high) frequency components are passed through the filter, that is, if $f_Y(\omega) = 0$ for $\omega \geq \omega_1$ ($\omega \leq \omega_1$) for some frequency ω_1 .

b) a bandpass filter if only frequency components in a certain interval (band) of frequencies are passed through the filter.

Thus the moving average smoother is an example of a low pass filter except that its frequency transfer function never becomes exactly zero for high frequencies. We can now see why $h(e^{2\pi i\omega})$ is called the frequency transfer function of the filter; it determines what happens to the various frequency components in X as a result of the filter.

Designing Filters

Sometimes we want to design filters, that is, determine filter coefficients so that an output time series Y has a certain type of spectral density. We illustrate this by considering the design of a bandpass filter. Suppose we want

$$f_Y(\omega) = \begin{cases} f_X(\omega), & \omega \in [u-v, u+v] \text{ or } \omega \in [1-(u+v), 1-(u-v)] \\ 0, & \text{otherwise;} \end{cases}$$

that is, we want to determine a bandpass filter centered at frequency u and having bandwidth v . Recall that a spectral density is symmetric about $\omega = .5$, that is, $f(\omega) = f(1-\omega)$ for $\omega \in [0, 1]$, and thus we must put both intervals in the definition of f_Y . Thus we are looking for a filter having frequency response function

$$h(e^{2\pi i\omega}) = \begin{cases} 1, & \omega \in [u-v, u+v] \text{ or } \omega \in [1-(u+v), 1-(u-v)] \\ 0, & \text{otherwise.} \end{cases}$$

This means that we must find (see Section A.2.1)

$$\begin{aligned} \beta_k &= \int_0^1 h(e^{2\pi i\omega}) e^{-2\pi i k \omega} d\omega \\ &= \int_{u-v}^{u+v} e^{-2\pi i k \omega} d\omega + \int_{1-(u+v)}^{1-(u-v)} e^{-2\pi i k \omega} d\omega. \end{aligned}$$

For $k = 0$, this gives $\beta_0 = 4v$, while for $k \neq 0$,

$$\beta_k = \frac{2}{\pi k} \cos 2\pi k u \sin 2\pi k v.$$

Thus the filter giving rise to the ideal transfer function h has infinitely many coefficients. In practice, we need to determine how many terms in the filter are needed to produce a satisfactory approximation to the transfer function. In the left column of graphs in Figure 2.4, we superimpose the transfer function h of the ideal bandpass filter for $u = .25$ and $v = .1$ and the M -term approximation

$$h_M(e^{2\pi i\omega}) = \sum_{k=-M}^M \beta_k e^{2\pi i k \omega}$$

to h for $M = 12, 24$, and 48 . Note that the approximation does not approach the ideal function in a smooth way; rather it is very wiggly and has a strange behavior at the points of discontinuity of h , namely at the values $.15$ and $.35$. This phenomenon of overshooting the function at a point of discontinuity is called Gibb's phenomenon in the theory of Fourier series. Note also that the approximating functions are negative in places, even though the true function is never negative.

The traditional method for eliminating Gibb's phenomenon and the negativity in approximating functions is to apply nonnegative weights to the β 's (the β 's are called the Fourier coefficients of h) before finding the approximating series. This is what we have done to obtain the second column of graphs in Figure 2.4. We have used the Parzen weights (see Table 3.3). Note how the approximating functions are approaching h much more smoothly and are never negative. The penalty for this is that for the same M , the unweighted function is closer to h as measured by the integral of the squared difference between h and h_M than is the function using the weights. See Example 2.4 for the macro that generated Figure 2.4.

The Lag Operator

Linear filters can be succinctly represented if we introduce what is called the lag (or backshift) operator.

Definition. The lag operator L operating on a time series X is defined by

$$L^k X(t) = X(t - k), \quad k \in \mathcal{Z}.$$

If Y is a filtered version of X we define the filter polynomial operator to be

$$h(L) = \sum_{k=-\infty}^{\infty} \beta_k L^k.$$

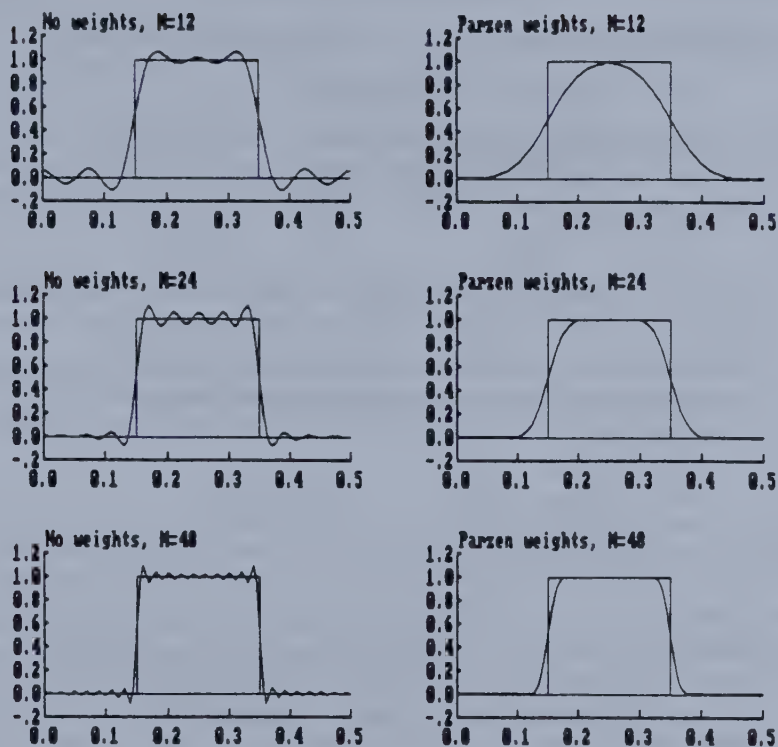


Figure 2.4. Approximations to the Ideal Bandpass Filter Transfer Function Using Unweighted and Weighted Fourier Coefficients.

Thus we can write formally

$$\begin{aligned}
 Y(t) &= \sum_{k=-\infty}^{\infty} \beta_k X(t-k) \\
 &= \sum_{k=-\infty}^{\infty} \beta_k L^k X(t) \\
 &= h(L)X(t).
 \end{aligned}$$

The FILT Command

As we saw in Section 1.5, the command

```
y=FILT(x,beta,beta0,n,m)
```

will form the array y by

$$y(i) = \sum_{j=0}^m \beta_j x(i+m-j), \quad i = 1, \dots, n-m.$$

Thus to generate a realization of length 100 from an MA(1) process having $\beta = .5$ and $\sigma^2 = 1$, and having exponential errors, we can use

```
beta=<.5>
e=WN(0,101,3)
x=FILT(e,beta,1,101,1)
```

2.4. Time Series Prediction

One of the major aims of time series analysis is to make use of correlation between and within series to predict unobserved values of a series. In this section we describe some of the basic theory for prediction. In Section 2.6 we discuss in detail several algorithms for calculating predictors for the time series models that we present in Section 2.5. Much of the theory of prediction is due to the remarkable work of Whittle (1963a).

If we have a realization $X(1), \dots, X(n)$ from a time series X , we often wish to find a function of the data that is close to some future variable $X(n+h)$. We call n the memory or origin of the predictor and h the horizon or number of steps ahead being predicted.

Definition. Let $\mathbf{X}_n = (X(1), \dots, X(n))^T$ be a realization of length n from a time series X .

a) The best unbiased predictor of $X(n+h)$ given \mathbf{X}_n is that function \tilde{X}_{nh} of \mathbf{X}_n that has the same mean as $X(n+h)$ and has smaller prediction error variance than any other unbiased function of \mathbf{X}_n .

b) The best unbiased linear predictor of $X(n+h)$ given \mathbf{X}_n is that linear function \hat{X}_{nh} of \mathbf{X}_n that has the same mean as $X(n+h)$ and has smaller prediction error variance than any other unbiased linear function of \mathbf{X}_n .

c) If \hat{X}_{nh} converges in mean square as $n \rightarrow \infty$ to a random variable X_{nh} , then X_{nh} is called the infinite memory, h step ahead predictor of $X(n+h)$.

d) The error variances of \tilde{X}_{nh} , \hat{X}_{nh} , and X_{nh} are denoted $\tilde{\sigma}_{nh}^2$, $\hat{\sigma}_{nh}^2$, and σ_{nh}^2 , respectively.

The following theorem is a straightforward application of the material in Section A.4.3 on prediction of random vectors. Before stating it we introduce a special matrix called the permutation matrix.

Definition. The $(n \times n)$ matrix \mathbf{P}_n consisting of all zeros except for ones on the main reverse diagonal is called the permutation matrix of order n .

We use the permutation matrix because many of the linear combinations that we use are of the form

$$l = a_1 x_n + a_2 x_{n-1} + \cdots + a_n x_1 = \mathbf{a}^T \mathbf{P}_n \mathbf{x},$$

where $\mathbf{a} = (a_1, \dots, a_n)^T$ and $\mathbf{x} = (x_1, \dots, x_n)$, since pre (post) multiplication of a matrix by the permutation matrix reverses the order of the rows (columns) of that matrix.

There are two other important features of the permutation matrix. First, $\mathbf{P}_n^2 = \mathbf{I}_n$, the $(n \times n)$ identity matrix. Second, if $\mathbf{\Gamma}_n$ is an $(n \times n)$ symmetric Toeplitz matrix, then (see Problem T2.13)

$$\mathbf{\Gamma}_n = \mathbf{P}_n \mathbf{\Gamma}_n \mathbf{P}_n \quad \text{and} \quad \mathbf{\Gamma}_n^{-1} = \mathbf{P}_n \mathbf{\Gamma}_n^{-1} \mathbf{P}_n.$$

With these facts in mind, we now present some of the basic facts about univariate time series prediction.

Theorem 2.4.1 UNIVARIATE PREDICTION

Let X be a zero mean time series. Then

a) The best unbiased predictor and its error variance are given by

$$\tilde{X}_{nh} = E(X(n+h) | X(1), \dots, X(n))$$

$$\tilde{\sigma}_{nh}^2 = \text{Var}(X(n+h) | X(1), \dots, X(n)).$$

b) If X is covariance stationary with autocovariance function R , then the best unbiased linear predictor and its error variance are given by

$$\hat{X}_{nh} = \boldsymbol{\lambda}_{nh}^T \mathbf{P}_n \mathbf{X}_n$$

$$\hat{\sigma}_{nh}^2 = R(0) - \mathbf{r}_{nh}^T \mathbf{\Gamma}_n^{-1} \mathbf{r}_{nh},$$

where $\mathbf{X}_n = (X(1), \dots, X(n))^T$ and the prediction coefficients

$$\lambda_{nh} = (\lambda_{nh}(1), \dots, \lambda_{nh}(n))^T$$

satisfy the prediction normal equations

$$\Gamma_n \lambda_{nh} = \mathbf{r}_{nh},$$

where $\Gamma_n = \text{Toepl}(R(0), \dots, R(n-1))$ and $\mathbf{r}_{nh} = (R(h), \dots, R(h+n-1))^T$. Further, these predictors and prediction error variances can be found using conditional means and variances as in part (a) but for a Gaussian time series having the same autocovariance function as X .

c) Let $\lambda_n(j)$ and $\hat{\sigma}_n^2$ denote the coefficients and prediction error variances for the best unbiased linear one step ahead predictor. Then the $\lambda_n(j)$ and $\hat{\sigma}_n^2$ satisfy Levinson's recursion:

$$\lambda_{j+1}(j+1) = \frac{R(j+1) - \sum_{k=1}^j \lambda_j(k)R(j+1-k)}{\hat{\sigma}_j^2}$$

$$\lambda_{j+1}(k) = \lambda_j(k) - \lambda_{j+1}(j+1)\lambda_j(j+1-k), \quad k = 1, \dots, j$$

$$\hat{\sigma}_{j+1}^2 = \hat{\sigma}_j^2(1 - \lambda_{j+1}^2(j+1)),$$

with $\lambda_1(1) = \rho(1)$ and $\hat{\sigma}_0^2 = R(0)$. Further, for $k > 1$, $\lambda_k(k)$ is equal to the correlation between the errors in predicting $X(t)$ from $X(t+1), \dots, X(t+k-1)$ and predicting $X(t+k)$ from $X(t+1), \dots, X(t+k-1)$. Thus $\lambda_k(k)$ is called the partial autocorrelation coefficient of lag k .

d) If X is covariance stationary and

$$\lim_{n \rightarrow \infty} \hat{\sigma}_{n1}^2 = \sigma_\infty^2 > 0,$$

then X is said to be purely nondeterministic and

i) There exists a white noise time series ϵ having variance σ_∞^2 and an infinite sequence of constants $\gamma_0 = 1, \gamma_1, \gamma_2, \dots$ such that, as a limit in mean square,

$$X(t) = \sum_{k=0}^{\infty} \gamma_k \epsilon(t-k).$$

This is called the infinite order moving average representation of X , and the ϵ 's are called the innovations of the process.

ii) The infinite memory predictor X_{nh} exists and is given by, as a limit in mean square,

$$X_{nh} = \sum_{k=h}^{\infty} \gamma_k \epsilon(n+h-k),$$

while

$$\sigma_{nh}^2 = \sigma_{\infty}^2 \sum_{k=0}^{h-1} \gamma_k^2.$$

Further, for $v \geq 0$,

$$\begin{aligned} \sigma_{hv} &= \text{Cov}(X_{nh}, X_{n,h+v}) \\ &= \sigma_{\infty}^2 \sum_{j=h}^{\infty} \gamma_j \gamma_{j+v} \\ &= R(v) - \sigma_{\infty}^2 \sum_{j=0}^{h-1} \gamma_j \gamma_{j+v}. \end{aligned}$$

iii) A sufficient condition for X to be purely nondeterministic is that X have a spectral density f satisfying

$$S = \int_0^1 \log f(\omega) d\omega > -\infty,$$

in which case

$$\sigma_{\infty}^2 = e^S.$$

e) WOLD DECOMPOSITION. Any covariance stationary time series X can be written as

$$X(t) = U(t) + V(t),$$

where U and V are uncorrelated with each other, U is purely nondeterministic, and V is purely deterministic.

Implications: Part (a) says that best predictors and their error variances are just conditional means and variances, while part (b) gives formulas for finding best linear predictors and their error variances. Note the similarity of the prediction normal equations to the normal equations in regression analysis, with $R(0)$ playing the role of $\mathbf{y}^T \mathbf{y}$, Γ_n playing the role of $\mathbf{X}^T \mathbf{X}$, and \mathbf{r}_{nh} playing the role of $\mathbf{X}^T \mathbf{y}$. The last part of part (b) provides an often used method for calculating linear predictors, namely the device of pretending that the process is Gaussian and then using conditional means and variances.

Part (c) of the theorem contains the often used Levinson algorithm (see Levinson (1947)). This algorithm allows us to find recursively the prediction coefficients. We will refer to this algorithm frequently in the rest of the book. The last part of part (c) shows that for each memory n , the last prediction coefficient is indeed a partial correlation. These partials are used extensively

in identifying time series models (see Section 3.4). In Section 2.6 we give more details about part (c) (see Theorems 2.6.3 and 2.6.4).

Part (d) of the theorem provides algorithms for finding infinite memory linear predictors. These are often very easy to calculate. Further, the infinite order MA representation of a process in terms of its innovations is often used in other contexts (see Theorem 2.6.8 for example). Note from part (ii) of (d) that the infinite memory one step ahead prediction error $X(n+1) - X_{n1} = \epsilon(n+1)$, which is the origin of the term *innovation*; that is, $\epsilon(n+1)$ is what is left over after having used the infinite past of X to predict $X(n+1)$. Finally, part (e) gives a way of decomposing any covariance stationary time series into two parts, one that can be perfectly predicted if we know enough of its past, and one part that cannot be perfectly predicted no matter how much of its past we know. Most of the time series models that we consider have only this unpredictable or nondeterministic part.

2.5. Time Series Models

In this section we present some of the models that have been used to represent data. The simplest model is the white noise process that we described in Section 2.3.

2.5.1. Random Walk Processes

The first process we discuss in this section is in fact not even covariance stationary, but occurs very frequently in the physical and economic sciences.

Definition. Suppose

$$X(t) = X(t-1) + \epsilon(t), \quad t \geq 1,$$

where $\epsilon \sim \text{WN}(\sigma^2)$. Then X is called a random walk process and we write $X \sim \text{RW}(\sigma^2)$.

Actually a random walk process is not fully specified until the characteristics of the starting value $X(0)$ are given. We usually assume that $X(0)$ is a random variable that is uncorrelated with any of the ϵ 's.

Theorem 2.5.1	PROPERTIES OF RANDOM WALKS
----------------------	----------------------------

Suppose $X \sim \text{RW}(\sigma^2)$ with

$$E(X(0)) = \mu_X, \quad \text{Var}(X(0)) = \sigma_X^2, \quad \text{Var}(\epsilon(t)) = \sigma_\epsilon^2, \quad \text{Cov}(X(0), \epsilon(t)) = 0.$$

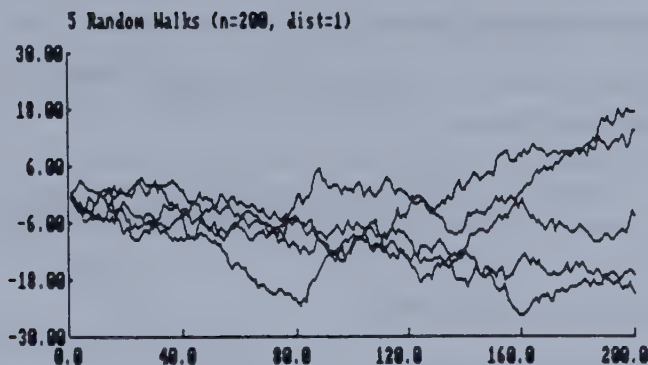


Figure 2.5. Five Realizations from a Gaussian Random Walk Process.

Then

$$E(X(t)) = \mu_X \quad \text{and} \quad \text{Var}(X(t)) = \sigma_X^2 + t\sigma_\epsilon^2, \quad t \geq 1.$$

Proof: We can write

$$\begin{aligned} X(t) &= X(t-1) + \epsilon(t) \\ &= [X(t-2) + \epsilon(t-1)] + \epsilon(t) \\ &= \dots \\ &= X(0) + \sum_{j=1}^t \epsilon(j), \end{aligned}$$

from which the results follow since $X(0)$ is uncorrelated with the ϵ 's.

Note that X is not covariance stationary since $\text{Var}(X(t))$ is not independent of t . Further, $\text{Var}(X(t)) \rightarrow \infty$ as $t \rightarrow \infty$. Figure 2.5 gives five realizations of length 200 from a Gaussian random walk process; that is, $X(0)$ and $\epsilon(1), \dots, \epsilon(200)$ are iid $N(0, 1)$ variables. As time progresses, the realizations get increasingly far apart. This is expected since $\text{Var}(X(t))$ is increasing linearly without bound as t increases. Note also that these realizations are similar in appearance to many price time series in economics such as stock market data. Finally, note that the first difference of a random walk process is a white noise process.

Further aspects of the random walk are considered in Problem T2.3, and Example 2.5 contains the macro that produced Figure 2.5.

Simulating Random Walks

A simple method for simulating random walks in TIMESLAB is

```
e=WN(seed,n,dist)
x=CUM(e,n,1)
```

since the **CUM** command finds the sequence of partial sums of an array, and if this is applied to white noise, one obtains a random walk process (see Problem T2.14).

Prediction of Random Walks

If $X \sim \text{RW}(\sigma^2)$, then

$$\begin{aligned}\tilde{X}_{n1} &= E[X(n+1)|X(1), \dots, X(n)] \\ &= E[X(n) + \epsilon(n+1)|X(1), \dots, X(n)] \\ &= E[X(n)|X(1), \dots, X(n)] + E[\epsilon(n+1)|X(1), \dots, X(n)] \\ &= X(n) + E[\epsilon(n+1)|X(1), \dots, X(n)].\end{aligned}$$

If this last conditional expectation is zero, we have that X is a Martingale process; that is, the best predictor of $X(n+1)$ given the previous n values is just the n th value. We have not required up to now that $X(0)$ be independent of the ϵ 's (which would make X a Martingale). If $X(0)$ and the ϵ 's are Gaussian, then of course we would get that $\tilde{X}_{n1} = X(n)$.

2.5.2. Harmonic Processes

Our second example is covariance stationary but has a spectral distribution function that is not absolutely continuous.

Definition. A time series X is called a harmonic process if

$$X(t) = \sum_{j=1}^M [A_j \cos 2\pi t \omega_j + B_j \sin 2\pi t \omega_j], \quad t \in \mathcal{Z},$$

where the A 's and B 's are zero mean, uncorrelated random variables having $\text{Var}(A_j) = \text{Var}(B_j) = \sigma_j^2$, and $\omega_1, \dots, \omega_M$ are distinct frequencies in $[0, .5]$.

For simplicity we will assume that $M = 1$ and thus drop the subscripts on the A 's, B 's, and σ^2 's while leaving the subscript on ω . Thus we have

$$E(X(t)) = E[A \cos 2\pi t \omega_1 + B \sin 2\pi t \omega_1] = 0,$$

while, since $E(AB) = 0$ and $E(A^2) = E(B^2) = \sigma^2$, we have

$$\begin{aligned}
 \text{Cov}(X(t), X(t+v)) &= E(X(t)X(t+v)) \\
 &= E[(A \cos 2\pi t\omega_1 + B \sin 2\pi t\omega_1)(A \cos 2\pi(t+v)\omega_1 + B \sin 2\pi(t+v)\omega_1)] \\
 &= \sigma^2 [\cos 2\pi t\omega_1 \cos 2\pi(t+v)\omega_1 + \sin 2\pi t\omega_1 \sin 2\pi(t+v)\omega_1] \\
 &= \sigma^2 \cos 2\pi v\omega_1,
 \end{aligned}$$

by the trigonometric identities

$$\cos(\alpha - \beta) = \cos \alpha \cos \beta + \sin \alpha \sin \beta \quad \text{and} \quad \cos(-\theta) = \cos \theta.$$

Thus $\text{Cov}(X(t), X(t+v))$ is independent of t and X is covariance stationary with

$$R(v) = \sigma^2 \cos 2\pi v\omega_1.$$

Note that $R(v)$ does not decay to zero as $v \rightarrow \infty$ since it is in fact periodic. Thus R is not absolutely summable. Recall that absolute summability is only a sufficient condition for the absolute continuity of a spectral distribution function (part (e) of Theorem 2.2.1). We can see that F has a jump discontinuity at $\omega = \omega_1$ if we define

$$F(\omega) = \begin{cases} 0, & \omega \in [0, \omega_1) \\ \frac{\sigma^2}{2}, & \omega \in [\omega_1, 1 - \omega_1) \\ \sigma^2, & \omega \in [1 - \omega_1, 1], \end{cases}$$

since it can be shown (see Section A.2.2) that $R(v) = \int_0^1 \cos 2\pi v\omega \, dF(\omega)$. Half of the variance is placed at ω_1 and half at $1 - \omega_1$ to satisfy the requirement that $F(\omega) = R(0) - F(1 - \omega)$ at all continuity points of F .

If we have a harmonic process with sinusoids of frequencies $\omega_1, \dots, \omega_M$, then F will have jumps of size $\sigma_1^2/2, \dots, \sigma_M^2/2$ at frequencies $\omega_1, \dots, \omega_M$. If $\omega \neq \omega_j$, then F is differentiable with derivative zero, while at $\omega = \omega_j$, the left derivative is infinite and the right derivative zero. Finally, note that a single realization from a harmonic process is indistinguishable from a deterministic sinusoid and getting more of a single realization provides no more information about the properties of the process.

2.5.3. Moving Average Processes

In our discussion of linear filters in Section 2.3 we introduced the moving average process:

$$X(t) = \sum_{k=0}^q \beta_k \epsilon(t-k), \quad \beta_0 = 1, \quad t \in \mathbb{Z},$$

where $\epsilon \sim \text{WN}(\sigma^2)$, and showed that its autocovariance function R and spectral density function f are given by

$$R(v) = \begin{cases} \sigma^2 \sum_{k=0}^{q-|v|} \beta_k \beta_{k+|v|}, & |v| \leq q \\ 0, & |v| > q \end{cases}$$

$$f(\omega) = \sigma^2 |h(e^{2\pi i \omega})|^2, \quad \omega \in [0, 1],$$

where the complex valued polynomial h is given by

$$h(z) = \sum_{k=0}^q \beta_k z^k.$$

If we write the process in operator notation as

$$X(t) = h(L)\epsilon(t),$$

it is of interest to determine whether we can invert the operator; that is, can we write in any sense $\epsilon(t) = (1/h(L))X(t)$? The answer is given by the following theorem.

Theorem 2.5.2	INVERTIBILITY OF MA PROCESSES
----------------------	--------------------------------------

Suppose $X \sim \text{MA}(q, \beta, \sigma^2)$. If the zeros of h are all greater than one in modulus, then

a) $f(\omega) > 0, \quad \omega \in [0, 1].$

b) We can write as a limit in mean square,

$$\epsilon(t) = \sum_{j=0}^{\infty} \alpha_j X(t-j), \quad \omega \in [0, 1],$$

where the α 's are the coefficients of the polynomial

$$g(z) = \frac{1}{h(z)} = \sum_{j=0}^{\infty} \alpha_j z^j$$

and are given by $\alpha_0 = 1$ and

$$\alpha_j = - \sum_{i=1}^{\min(q,j)} \beta_i \alpha_{j-i}, \quad j \geq 1.$$

Proof: Part (a) follows because

$$|e^{2\pi i \omega}|^2 = |\cos 2\pi \omega + i \sin 2\pi \omega|^2 = \cos^2 2\pi \omega + \sin^2 2\pi \omega = 1,$$

and thus the condition that $h(z) \neq 0$ unless $|z| > 1$ means that $f(\omega) = \sigma^2 |h(e^{2\pi i \omega})|^2 > 0$.

Rather than prove part (b) in general, we illustrate its proof by considering the simple MA(1) process $X(t) = \epsilon(t) + \beta \epsilon(t-1)$. Successively substituting for $\epsilon(t-1), \epsilon(t-2), \dots$ gives

$$\begin{aligned} \epsilon(t) &= X(t) - \beta \epsilon(t-1) \\ &= X(t) - \beta [X(t-1) - \beta \epsilon(t-2)] \\ &= X(t) - \beta X(t-1) + \beta^2 [X(t-2) - \beta \epsilon(t-3)] \\ &= \dots \\ &= \sum_{j=0}^K (-\beta)^j X(t-j) + (-\beta)^{K+1} \epsilon(t-K+1), \end{aligned}$$

which gives

$$\begin{aligned} E \left(\epsilon(t) - \sum_{j=0}^K (-\beta)^j X(t-j) \right)^2 &= E [(-\beta)^{2K+2} \epsilon^2(t-K+1)] \\ &= (-\beta)^{2K+2} \sigma^2 \rightarrow 0 \end{aligned}$$

as $K \rightarrow \infty$, if $|\beta| < 1$. But $h(z) = 1 + \beta z$ has the zero $z = -1/\beta$ which is greater than one in modulus (in this case absolute value since z is real) if and only if $|\beta| < 1$. Note that the general expression for α_j does in fact give $\alpha_j = (-\beta)^j$ as required.

We can now formalize our definition of an invertible MA process.

Definition. An $MA(q, \beta, \sigma^2)$ process is said to be invertible if all of the zeros of its characteristic polynomial h are greater than one in modulus.

Nonidentifiability of MA Processes

If we are given the parameters β and σ^2 of an $MA(q)$ process, then we can calculate the corresponding autocovariances by

$$R(v) = \begin{cases} \sigma^2 \sum_{k=0}^{q-|v|} \beta_k \beta_{k+|v|}, & |v| \leq q \\ 0, & |v| > q. \end{cases}$$

On the other hand, if we are given q and the first $q + 1$ autocovariances of the process, there is no unique set of parameters β and σ^2 giving rise to these covariances. This is called the nonidentifiability of an MA process. For example, for an $MA(1)$ process, the autocovariances $R(0) = 5$ and $R(1) = 2$ could have arisen from $\sigma^2 = 1$ and $\beta_1 = 2$ or from $\sigma^2 = 4$ and $\beta_1 = .5$. For a general $MA(q, \beta, \sigma^2)$ process, there are $2^q - 1$ other sets of parameters all leading to the same autocovariance function as do β and σ^2 . These other sets of parameters can be determined by (1) finding the zeros of the characteristic polynomial corresponding to β , and then (2) constructing the other polynomials having combinations of these zeros and their reciprocals. Thus for an $MA(2)$ process, we find the zeros z_1 and z_2 of the characteristic polynomial, and then find the three polynomials having zeros z_1 and $1/z_2$, $1/z_1$ and z_2 , and finally $1/z_1$ and $1/z_2$, in each case finding the value of the error variance that makes the autocovariance function agree with the one corresponding to the original parameters. In Section 2.6.3 we consider methods for finding values of β and σ^2 for a given autocovariance function. The values that will be found are those whose characteristic polynomial has all of its zeros outside the unit circle.

Commands for MA Processes

There are several commands that are useful for studying MA processes. First, **MACORR** and **MASP** calculate ρ and f from an MA process given its parameters. The command **MDT** simulates data from an MA process having $N(0, 1)$ errors. To get data from an MA having nonnormal errors, the **WN** and **FILT** commands can be combined. The **CORRMA** command (see Section 2.6) uses an algorithm developed by Wilson (1979) to find the β 's corresponding to a given set of correlations.

Prediction for MA Processes

In Section 2.6, we derive general expressions for the predictors and prediction error variances for an MA process. These expressions will be functions of the elements of the modified Cholesky decomposition (see Section A.1.1) of the covariance matrix of the MA process. A distinguishing feature of this covariance matrix is that it is banded; that is, its elements outside of a band formed by its main diagonal and the q diagonals on either side are zero. This leads to simple forms for the predictors. For now we note that for a Gaussian MA process,

$$\begin{aligned}\tilde{X}_{nh} &= E(X(n+h)|\mathbf{X}_n) \\ &= \sum_{k=0}^q \beta_k E(\epsilon(n+h-k)|\mathbf{X}_n) \\ &= 0, \quad h > q,\end{aligned}$$

since $\epsilon(t)$ is independent of $X(s)$ for $t > s$. Thus by the device described in part (b) of Theorem 2.4.1, we can write the following recursion for the best linear predictors:

$$\hat{X}_{nh} = \begin{cases} \sum_{k=0}^q \beta_k \hat{X}_{n,h-k}, & h = 1, \dots, q \\ 0, & h > q. \end{cases}$$

Partial Autocorrelations for MA Processes

In Theorem 2.4.1 we saw that the partial autocorrelation of lag v of a time series X is the last coefficient $\lambda_v(v)$ in the best v step ahead linear predictor. In Section 2.5.4 we will describe a simple method for calculating these partial autocorrelations. For now we note that unlike the ordinary autocorrelations which become identically zero for lags greater than the order q , the partial autocorrelations of an MA process decay to zero exponentially (see the examples at the end of this section).

MA Spectra and Trigonometric Polynomials

Before leaving MA processes, we note that since $R(v) = 0$ for $|v| > q$, we can write from part (f) of Theorem 2.2.1,

$$f(\omega) = R(0) + 2 \sum_{v=1}^q R(v) \cos 2\pi v\omega.$$

Now we can also write $\cos 2\pi v\omega$ as a polynomial of degree v in $\cos 2\pi\omega$ since for $v \geq 2$, we have the important trigonometric identity

$$\cos v\theta = 2 \cos \theta \cos(v-1)\theta - \cos(v-2)\theta,$$

which if used recursively, ultimately expresses $\cos v\theta$ as a polynomial in $\cos \theta$. Thus the spectral density of an $\text{MA}(q)$ can be written as a q th-degree trigonometric polynomial. The above identity is also important in other contexts in time series analysis. If we write $z(v) = \cos v\theta$, we have

$$z(v) - 2 \cos \theta z(v-1) + z(v-2) = 0, \quad v \geq 2,$$

with $z(0) = 1$ and $z(1) = \cos \theta$, which is a second-order difference equation with initial conditions $z(0)$ and $z(1)$. This equation has solution $z(v) = \cos v\theta$ (see Problem T1.7).

Examples of MA Processes

In this section we have seen:

1. The autocorrelation function for an $\text{MA}(q)$ process is identically zero for lags greater than the order q .
2. The partial autocorrelation function decays to zero as v increases.
3. The spectral density function of an $\text{MA}(q)$ process is a q th-degree trigonometric polynomial. Thus for small values of q , it is difficult for the spectral density to have sharp peaks.

In Figure 2.6 we display the correlogram, partial correlogram, spectral density, and one realization of length 200 from each of two MA processes. The two processes are:

$$\text{Model 1: } X(t) = \epsilon(t) - 0.70\epsilon(t-1) - 0.10\epsilon(t-2) + 0.60\epsilon(t-3)$$

$$\text{Model 2: } X(t) = \epsilon(t) + 0.80\epsilon(t-4);$$

that is, the first process is an $\text{MA}(3)$ process while the second is a (subset) $\text{MA}(4)$. In both cases, the correlogram is zero for lags greater than the order of the process, while the partial correlogram is not. The partials for the subset model are zero except for lags that are multiples of four. See Example 2.7 for the macro that generated this figure.

2.5.4. Autoregressive Processes

Perhaps the most often used time series model in practice is the autoregressive process. Intuitively, X is an autoregressive process of order p with coefficients $\alpha_1, \dots, \alpha_p$ and noise variance σ^2 if it can be transformed to white noise by a filter of length p ; that is,

$$X(t) + \alpha_1 X(t-1) + \dots + \alpha_p X(t-p) = \epsilon(t), \quad t \in \mathcal{Z},$$

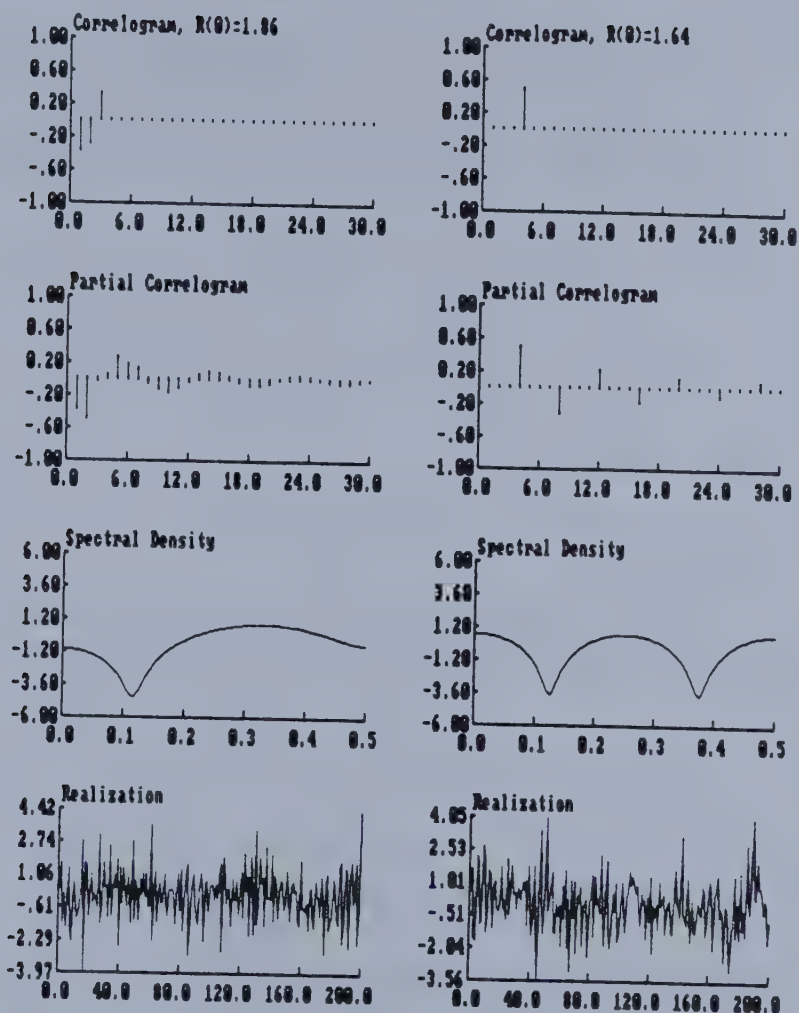


Figure 2.6. Correlogram, Partial Correlogram, Spectral Density, and a Realization of Length 200 for Each of Two MA Processes.

where $\epsilon \sim WN(\sigma^2)$. One appeal of this process is that it is in the form of a regression model

$$X(t) = -\alpha_1 X(t-1) - \dots - \alpha_p X(t-p) + \epsilon(t),$$

with past (or "lagged") values of X as the independent variables in the regression.

It is important to note that the autoregressive process is not defined explicitly, but rather as the solution to a stochastic difference equation, that is, a difference equation whose terms are random variables. The difficulty with this is in determining whether there is in fact a covariance stationary time series that satisfies such an equation. To illustrate this, suppose that X satisfies the difference equation with $p = 1$:

$$X(t) + \alpha X(t-1) = \epsilon(t).$$

Successively substituting for $X(t-1), X(t-2), \dots$ gives

$$X(t) - \sum_{j=0}^{K-1} (-\alpha)^j \epsilon(t-j) = -(\alpha)^K X(t-K).$$

If $|\alpha| < 1$, then taking the limit of the expected value of the square of both sides of this results in

$$X(t) = \sum_{j=0}^{\infty} \beta_j \epsilon(t-j),$$

where $\beta_j = (-\alpha)^j$.

Now we could also write $\frac{1}{\alpha}X(t+1) + X(t) = \frac{1}{\alpha}\epsilon(t+1)$, which gives upon successive substitution

$$\begin{aligned} X(t) &= -\frac{1}{\alpha}[X(t+1) + \epsilon(t+1)] \\ &= -\frac{1}{\alpha}\left[-\frac{1}{\alpha}(X(t+2) + \epsilon(t+2)) + \epsilon(t+1)\right] \\ &= \dots \\ &= \left(-\frac{1}{\alpha}\right)^K X(t+K) + \sum_{j=1}^K \left(-\frac{1}{\alpha}\right)^j \epsilon(t+j), \end{aligned}$$

which, if $|\alpha| > 1$, gives as a limit in mean square

$$X(t) = \sum_{j=1}^{\infty} \beta_j \epsilon(t+j),$$

where $\beta_j = (-1/\alpha)^j$. If $|\alpha| = 1$ we are not able to do this process as neither $(-\alpha)^K$ nor $(-1/\alpha)^K$ goes to zero.

These arguments can be extended to $p > 1$ where instead of the location of $|\alpha|$ determining whether to write $X(t)$ as a function of past and present or future ϵ 's, the location of the zeros of the complex valued polynomial

$$g(z) = \sum_{j=0}^p \alpha_j z^j, \quad \alpha_0 = 1,$$

determines the representation. The following theorem summarizes the basic results.

Theorem 2.5.3 PROPERTIES OF AUTOREGRESSIVE PROCESSES

Let $\alpha_0 = 1$ and $\alpha_1, \dots, \alpha_p$ be a set of real constants, and define the complex valued polynomial (called the characteristic polynomial of the stochastic difference equation) $g(z) = \sum_{j=0}^p \alpha_j z^j$. Let $\epsilon \sim \text{WN}(\sigma^2)$. Then

a) If none of the zeros of g are equal to one in modulus, then

$$X(t) = \sum_{j=-\infty}^{\infty} \beta_j \epsilon(t-j), \quad t \in \mathcal{Z},$$

exists as a limit in mean square where the β 's are the coefficients of the polynomial

$$h(z) = \frac{1}{g(z)}.$$

This representation is called the (doubly) infinite order moving average representation of X . Further, X satisfies the autoregressive difference equation, is covariance stationary, has spectral density

$$f(\omega) = \sigma^2 \frac{1}{|g(e^{2\pi i \omega})|^2} = \sigma^2 |h(e^{2\pi i \omega})|^2,$$

and has autocovariance function R satisfying

$$\sum_{j=0}^p \alpha_j R(j-v) = \sigma^2 \beta_{-v}, \quad v \in \mathcal{Z}.$$

b) If the zeros of g are all less than one in modulus, then $\beta_j = 0$ for $j \geq 0$; that is, $X(t)$ can be expressed as a function of ϵ 's at only future times.

c) If the zeros of g are all greater than one in modulus, then $\beta_j = 0$ for $j < 0$; that is, $X(t)$ can be expressed as a function of ϵ 's at only the present and past times. Thus the equations relating the R and the α 's can be written as

$$\sum_{j=0}^p \alpha_j R(j-v) = \delta_v \sigma^2, \quad v \geq 0,$$

which are called the Yule-Walker equations.

d) If any of the zeros of g are equal to one in modulus, then there is no $MA(\infty)$ representation of X , and in fact there is no covariance stationary solution to the autoregressive difference equation. However, if one specifies the distribution of any p consecutive X 's, one can generate a (nonstationary) time series satisfying the difference equation and agreeing with the specified initial conditions.

Proof: The results in parts (a)–(c) about the coefficients of $1/g(z)$ follow from properties of Laurent expansions of analytic functions (see Ahlfors (1953)). The fact that the infinite order MA process satisfies the difference equation is verified by some simple algebra. We can then use the Filter Theorem to say that X is covariance stationary and that the spectral densities f_ϵ and f of ϵ and X are related by

$$\sigma^2 = f_\epsilon(\omega) = |g(e^{2\pi i\omega})|^2 f(\omega),$$

from which we obtain

$$f(\omega) = \frac{\sigma^2}{|g(e^{2\pi i\omega})|^2}.$$

To verify the last equation in part (a), we multiply both sides of the autoregressive difference equation by

$$X(t-v) = \sum_{k=-\infty}^{\infty} \beta_k \epsilon(t-v-k)$$

and take expected values of both sides, which gives

$$\begin{aligned} \sum_{j=0}^p \alpha_j R(j-v) &= E \left(\epsilon(t) \sum_{k=-\infty}^{\infty} \beta_k \epsilon(t-v-k) \right) \\ &= \sigma^2 \sum_{k=-\infty}^{\infty} \beta_k \delta_{k+v} \\ &= \sigma^2 \beta_{-v}. \end{aligned}$$

Part (d) is just a generalization of the random walk process. Given initial conditions, we can generate all the rest of the X 's from the ϵ 's via the difference equation.

From this theorem we can see that as long as none of the zeros of g are on the unit circle (that is, equal to one in modulus), the infinite order moving average process provides a covariance stationary solution to the stochastic difference equation and has the properties given by the theorem. If all of the

zeros of g are outside the unit circle, then Problem T2.15 describes a method for explicitly defining a stationary solution.

Since in practice we do not observe the future, we would like to define an autoregressive process so that $X(t)$ is only a function of $\{\epsilon(s), s \leq t\}$. We formalize this in the following definition.

Definition. A time series X is said to be an autoregressive process of order p with coefficients $\alpha = (\alpha_1, \dots, \alpha_p)^T$ and noise variance σ^2 if X satisfies the stochastic difference equation

$$X(t) + \alpha_1 X(t-1) + \dots + \alpha_p X(t-p) = \epsilon(t)$$

where the zeros of $g(z) = \sum_{j=0}^p \alpha_j z^j$ are all greater than one in modulus. We denote such a time series by $X \sim \text{AR}(p, \alpha, \sigma^2)$.

Recall for an MA(q) process that $R(v) = 0$ for $|v| > q$; that is, as soon as the lag v gets larger than the order of the process, $X(t)$ and $X(t+v)$ are no longer correlated. The autoregressive process has a correlation function that often gives a more realistic description of how $X(t)$ and $X(t+v)$ become uncorrelated as v increases. We will illustrate this by first considering the AR(1) and AR(2) cases and then presenting a general result about the behavior of the autocorrelation function in Theorem 2.5.4. For an AR(1) process having coefficient α and error variance σ^2 , we can write the Yule-Walker equations for $v = 0$ and $v = 1$ as

$$R(0) + \alpha R(1) = \sigma^2$$

$$R(1) + \alpha R(0) = 0,$$

since $R(1) = R(-1)$. From the second equation we have $R(1) = -\alpha R(0)$, which when substituted into the first equation gives

$$R(0) = \frac{\sigma^2}{1 - \alpha^2}.$$

From the Yule-Walker equations for $v > 0$, we have

$$R(v) = -\alpha R(v-1) = (-\alpha)^2 R(v-2) = \dots = (-\alpha)^v R(0),$$

which gives

$$R(v) = \frac{(-\alpha)^v \sigma^2}{1 - \alpha^2}, \quad \rho(v) = \frac{R(v)}{R(0)} = (-\alpha)^v, \quad v \geq 0.$$

Finally, since $R(v) = R(-v)$, we have

$$R(v) = \frac{(-\alpha)^{|v|}\sigma^2}{1-\alpha^2}, \quad \rho(v) = (-\alpha)^{|v|}, \quad v \in \mathcal{Z}.$$

Note that a sequence $\{a(v), v \geq 0\}$ is said to decay exponentially to zero if we can write

$$|a(v)| < c\delta^v,$$

where $|\delta| < 1$. Thus the autocovariance function (and hence the autocorrelation function) of an AR(1) process decays exponentially to zero. The rate of decay depends on how close α is to one.

For an AR(2) process, we have for $v > 2$,

$$R(v) + \alpha_1 R(v-2) + \alpha_2 R(v-2) = 0,$$

which is a second order homogeneous difference equation. From Theorem 1.6.1, we know that $R(v)$, which is the solution to the difference equation, can only be one of three forms depending on the nature of the zeros z_1 and z_2 of the polynomial $z^2 + \alpha_1 z + \alpha_2$. If they are real and distinct, then $R(v) = \beta_1 z_1^v + \beta_2 z_2^v$; if they are real and equal, then $R(v) = (\beta_1 + \beta_2 v) z_1^v$; and if they are complex conjugate pairs, we have $R(v) = \gamma \cos(v\theta + \delta) |z_1|^v$. In all three cases, R will decay to zero exponentially since the zeros of $z^2 + \alpha_1 z + \alpha_2$ will be less than one in modulus since they are the reciprocals of the zeros of $1 + \alpha_1 z + \alpha_2 z^2$ which are outside the unit circle. Here the rate of decay depends on how close to the unit circle is the zero that is closer to it. If the zeros are complex, the autocorrelation function will appear sinusoidal as well as decaying exponentially.

The next theorem shows that this exponential decay of ρ is true for every AR order.

Theorem 2.5.4 EXPONENTIAL DECAY OF AR CORRELATIONS

Suppose $X \sim \text{AR}(p, \alpha, \sigma^2)$ and let R be the autocovariance function of X . Let z_1, \dots, z_p be the zeros of $g(z)$. Then

$$|R(v)| < \kappa \left(\max_i \frac{1}{|z_i|} \right)^v$$

for some constant $\kappa > 0$.

For details on the proof of this theorem, see Anderson (1971), p. 175. Note that the theorem says that $R(v)$ is an exponentially decaying function of the reciprocal of the modulus of the zero of g that is closest to the unit circle. If a zero is close to the unit circle, $R(v)$ may decay very slowly.

Autoregressive processes are very popular also because as we will see in Chapter 3, the estimation of their parameters and the forecasting of future values are rather easy from a computational point of view. Further, the following theorem (see Fuller (1976), p. 150) shows that a wide variety of covariance stationary time series can be arbitrarily well approximated by an AR process.

Theorem 2.5.5 **AUTOREGRESSIVE APPROXIMATION**

Let X be a covariance stationary time series having spectral density f_X . Then for any $\delta > 0$, there exists a time series Y having spectral density function f_Y such that

$$|f_X(\omega) - f_Y(\omega)| < \delta, \quad \omega \in [0, 1],$$

while Y satisfies

$$\sum_{j=0}^p \alpha_j Y(t-j) = \epsilon(t),$$

where p is a finite integer, $\alpha_0 = 1$, the zeros of $\sum_{j=0}^p \alpha_j z^j$ are all outside the unit circle, and $\epsilon \sim \text{WN}(\sigma^2)$.

Prediction for AR Processes

Prediction is very simple for AR processes as we can write for a Gaussian $\text{AR}(p)$ process

$$\begin{aligned} \tilde{X}_{nh} &= E(X(n+h)|\mathbf{X}_n) \\ &= E(\epsilon(n+h)|\mathbf{X}_n) - \sum_{j=1}^p \alpha_j E(X(n+h-1)|\mathbf{X}_n) \\ &= - \sum_{j=1}^p \alpha_j \tilde{X}_{n,h-j} \end{aligned}$$

since $\epsilon(t)$ is independent of $X(s)$ for $t > s$. Note that for $j \geq h$, $\tilde{X}_{n,h-j} = X(n+h-j)$ since then $X(n+h-j)$ is an element of the set being conditioned upon. Thus the AR predictors satisfy the homogeneous difference equation

$$\sum_{j=0}^p \alpha_j \tilde{X}_{n,h-j} = 0.$$

Further

$$\text{Var}(\tilde{X}_{n,h} - X(n+h)) = \sigma^2 \sum_{k=0}^{h-1} \beta_k^2,$$

where the β 's are the coefficients of the $MA(\infty)$ representation of X .

From these facts, we also obtain an important fact about the partial autocorrelations θ of an $AR(p)$ process, namely that for any lag v greater than the order p , we have that $\theta_v = 0$.

Commands for AR Processes

TIMESLAB has several commands for studying AR processes. The algorithms used for several of the commands are described in detail in Section 2.6. First, ARDT will simulate a realization from a Gaussian AR process. To generate a realization of length n from an AR process having non-Gaussian errors, one can use code of the following form:

```
n2=2*n
ep=line(p,0,0)
e=WN(seed,n2,dist)
e=<ep,e>
x=ARDT(alpha,p,n2,e)
np1=n+1
x=EXTRACT(x,np1,n2)
```

where `dist` is the code for the distribution desired for the errors. Note that this uses zeros as starting values in the difference equation and then lets the recursion run for twice the length of the desired data set, and extracts the second half of the realization to be used as the data set. The effect of the starting values should be eliminated in this way.

The `ARCORR` and `CORRAR` commands find correlations from parameters and vice versa. The `ARSP` command calculates the spectral density function of an AR process from its parameters, while `ARPART` and `PARTAR` do the transformations from AR parameters to partial autocorrelations and back again. Note that `ARDT`, `ARCORR`, and `ARPART` all check whether the zeros of the characteristic polynomial are all outside the unit circle.

The command `DTAR` is a very general command that can be used to identify AR orders and estimate AR parameters by a variety of methods. See Section 3.5 for the methods that are used in the command.

Examples of AR Processes

We have now seen that for an $AR(p)$ process:

1. The autocorrelation function satisfies a difference equation of order p (namely the Yule-Walker Equations) and thus, since the zeros of the characteristic polynomial are all outside of the unit circle, the correlogram decays exponentially to zero.

2. The partial autocorrelation function is identically zero for lags greater than p .
3. The spectral density is the reciprocal of a p th degree trigonometric polynomial. This means that it is capable of having very sharp peaks.

In Figure 2.7 we give the correlogram, partial correlogram, spectral density, and a realization of length 200 for each of the two AR models:

$$\text{Model 1: } X(t) - 0.80X(t-1) + 0.40X(t-2) = \epsilon(t)$$

$$\text{Model 2: } X(t) - 0.90X(t-1) + 0.70X(t-2) = \epsilon(t).$$

Note how the qualitative features of the data correspond to the spectra and correlations. In particular, notice that the correlograms are of two types. In Series 1, the correlogram decays relatively rapidly to 0, while in Series 2, the correlogram follows a sinusoidal decay. See Example 2.7 for the macro that generates the graphs in this figure.

Periodically Correlated AR Processes

A common occurrence in seasonal data is that each $X(t)$ is linearly related to previous values of X , but that the relationship (as expressed by the number of previous values or their coefficients) varies from one season to another. A model that has been used to reflect this possibility is the periodic autoregressive model (see Jones and Brelsford (1967), Pagano (1978), and Newton (1982)). In the definition that follows we will assume that the X 's have been arranged so that $X(1)$ corresponds to the first of d "seasons."

Definition. A time series X is said to be a periodic autoregressive process of period d , orders p_1, \dots, p_d , coefficients $\alpha_k(j)$, and residual variances $\sigma_1^2, \dots, \sigma_d^2$ if

$$\sum_{j=0}^{p_k} \alpha_k(j) X(t-j) = \epsilon(t), \quad t \in \mathcal{Z},$$

where $k = \text{mod}(t-1, d) + 1$, and the ϵ 's are zero mean, uncorrelated random variables with $\text{Var}(\epsilon(t)) = \sigma_k^2$.

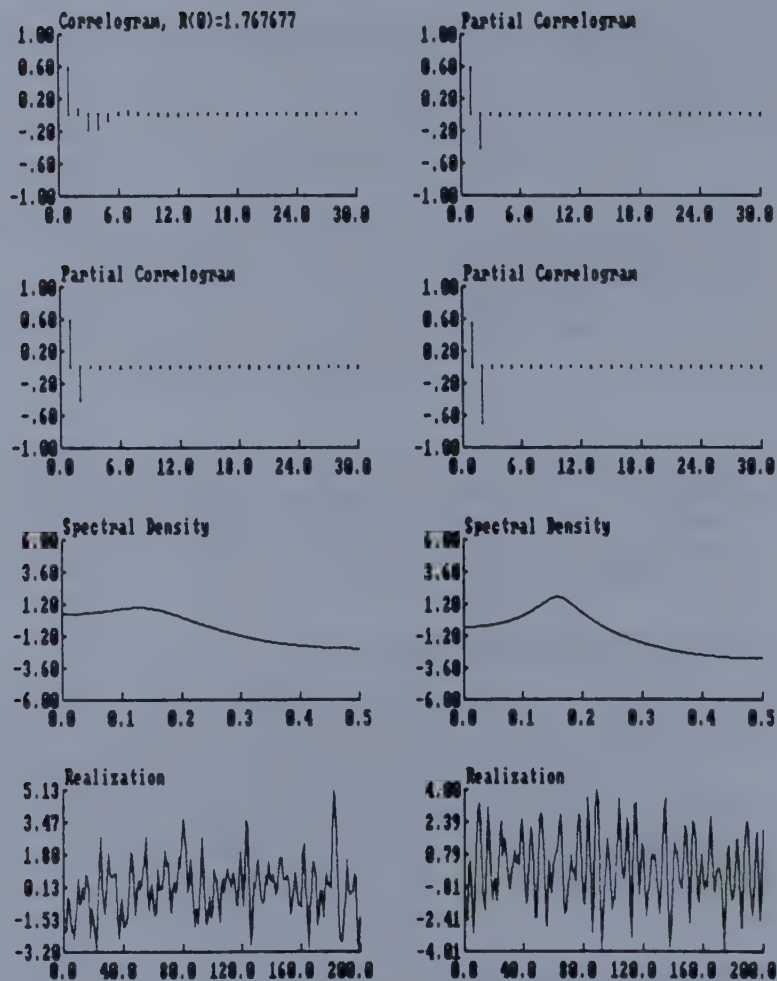


Figure 2.7. Correlogram, Partial Correlogram, Spectral Density, and a Realization of Length 200 for Each of Two AR Processes.

2.5.5. Autoregressive-Moving Average Processes

In the previous two sections we studied MA and AR processes. In this section we consider the autoregressive-moving average (ARMA) process which in a variety of ways is a combination of the AR and MA processes.

Definition. A time series X is said to be an autoregressive-moving average process of orders p and q , coefficients $\alpha = (\alpha_1, \dots, \alpha_p)^T$ and $\beta = (\beta_1, \dots, \beta_q)^T$, and noise variance σ^2 if X satisfies

$$\sum_{j=0}^p \alpha_j X(t-j) = \sum_{k=0}^q \beta_k \epsilon(t-k), \quad t \in \mathcal{Z},$$

where $\epsilon \sim \text{WN}(\sigma^2)$ and the zeros of the complex polynomial $g(z) = \sum_{j=0}^p \alpha_j z^j$ are all outside the unit circle. We denote such a time series by

$$X \sim \text{ARMA}(p, q, \alpha, \beta, \sigma^2).$$

If the zeros of the complex polynomial $h(z) = \sum_{k=0}^q \beta_k z^k$ are all outside the unit circle, then we say that X is invertible.

The operators $g(L)$ and $h(L)$ are called the AR and MA operators, respectively. Using arguments similar to those given in the previous two sections, we can show that an ARMA process has the $\text{MA}(\infty)$ representation

$$X(t) = \sum_{k=0}^{\infty} \gamma_k \epsilon(t-k), \quad t \in \mathcal{Z},$$

where the γ_k 's are the coefficients of the polynomial $h(z)/g(z)$, while if X is invertible, it has the $\text{AR}(\infty)$ representation

$$\sum_{j=0}^{\infty} a_j X(t-j) = \epsilon(t), \quad t \in \mathcal{Z},$$

where the a_j 's are the coefficients of the polynomial $g(z)/h(z)$. Thus we are able to justify the idea of writing $g(L)X(t) = h(L)\epsilon(t)$ as

$$X(t) = \frac{h(L)}{g(L)} \epsilon(t)$$

or

$$\frac{g(L)}{h(L)} X(t) = \epsilon(t).$$

In the next theorem, we express the autocovariance and spectral density functions of an ARMA process in terms of its parameters.

Theorem 2.5.6

 R AND f FOR AN ARMA PROCESS

If

$$X \sim \text{ARMA}(p, q, \alpha, \beta, \sigma^2),$$

then

a) $f(\omega) = \sigma^2 \frac{|h(e^{2\pi i \omega})|^2}{|g(e^{2\pi i \omega})|^2}.$

b) The autocovariances R satisfy

$$\begin{aligned} \sum_{j=0}^p \alpha_j R(j-v) &= \sum_{k=0}^q \beta_k R_{X\epsilon}(v-k), & v \geq 0 \\ &= 0, & v > q, \end{aligned}$$

where the cross-covariances $R_{X\epsilon}$ are given by

$$R_{X\epsilon}(v) = \text{Cov}(X(t), \epsilon(t+v)) = \begin{cases} 0, & v > 0 \\ \sigma^2 \gamma_{-v}, & v \leq 0 \end{cases}$$

and the γ_v 's are the coefficients of the $\text{MA}(\infty)$ representation of X .

c) The covariance generating function Γ_X of X is given by

$$\Gamma_X(z) = \sigma^2 \frac{h(z)h(z^{-1})}{g(z)g(z^{-1})},$$

where h and g are the characteristic polynomials for the MA and AR parts of the model, respectively.

Proof: To prove part (a), define the time series Y by $Y(t) = g(L)X(t) = h(L)\epsilon(t)$, and note that since Y is a filtered version of ϵ , we know that it is covariance stationary and we can write $f_Y(\omega) = \sigma^2 |h(e^{2\pi i \omega})|^2$. Further, since the zeros of g are all outside the unit circle, we know that X could be written as a filtered version of the covariance stationary series Y and thus is itself stationary, and $f_Y(\omega) = |g(e^{2\pi i \omega})|^2 f_X(\omega)$. Equating the two expressions for $f_Y(\omega)$ and solving for $f_X(\omega)$ give part (a).

Multiplying both sides of the ARMA difference equation by $X(t-v)$ and taking expected values give

$$\sum_{j=0}^p \alpha_j \text{E}(X(t-j)X(t-v)) = \sum_{k=0}^q \beta_k \text{E}(\epsilon(t-k)X(t-v))$$

Table 2.1. Nature of R and f for ARMA Processes

Process	$R(v)$ or $\rho(v)$	$f(\omega)$
MA(q)	0 for $ v > q$	q th-degree trigonometric polynomial
AR(p)	exponential decay for $v > 0$	reciprocal of p th-degree trigonometric polynomial
ARMA(p, q)	exponential decay for $v > q$	(q, p) th-degree rational trigonometric polynomial

or

$$\sum_{j=0}^p \alpha_j R(j-v) = \sum_{k=0}^q \beta_k R_{X\epsilon}(v-k).$$

Since $X(t)$ can be expressed in terms of $\epsilon(s)$ for $s \leq t$, we have that $X(t)$ is uncorrelated with future ϵ 's; that is, $R_{X\epsilon}(v) = 0$ for $v > 0$. For $v \leq 0$, we have

$$R_{X\epsilon}(v) = \sum_{k=0}^{\infty} \gamma_k E(\epsilon(t-k)\epsilon(t+v)),$$

and all of these expectations are zero except when $k = -v$.

Thus R satisfies a homogeneous difference equation of order p for $v > q$. The equations for $v = q+1, \dots, q+p$ are often called the high order Yule-Walker equations. We can now summarize the nature of R (or ρ) and f for AR, MA, and ARMA processes as in Table 2.1.

Prediction for ARMA Processes

In Section 2.6 we derive algorithms for finding predictors and their error variances for ARMA processes from two points of view. In Section 3.4.2 we describe a method for finding approximate predictors and variances.

Commands for ARMA Processes

The **ARMADT** command generates a realization from a Gaussian ARMA process, while to get a realization from an ARMA process having non-Gaussian errors, one can use the **ARDT** command in a similar fashion to generating non-Gaussian AR data except that the forcing term in the difference equation is a realization from a non-Gaussian MA process rather than a non-Gaussian WN process (see Problem C2.8).

The **ARMASP** command calculates the spectral density of an ARMA process, and the **ARMACORR** and **CORRARMA** commands find correlations from parameters

and vice versa. The **ARMAPRED** command finds predictors and prediction error variances for ARMA data. The algorithms for these last three commands are described in Section 2.6. To estimate parameters of ARMA schemes, we describe in Chapter 3 the **ARMASEL**, **DTARMA**, and **SEASEST** commands.

Examples of ARMA Processes

As we have seen, the ARMA process is a combination of the AR and MA processes. The basic characteristics of an ARMA model are:

1. Neither the correlogram nor the partial correlogram is identically zero past some lag; rather they each decay exponentially past some lag.
2. The spectral density function is the ratio of a q th-degree trigonometric polynomial to a p th-degree trigonometric polynomial. Thus it can have sharp peaks and/or sharp troughs.

In Figure 2.8 we give the correlogram, partial correlogram, spectral density, and a realization of length 200 from each of the two ARMA models:

$$1: \quad X(t) - 1.20X(t-1) + 0.78X(t-2) = \epsilon(t) + 0.45\epsilon(t-1) + 0.39\epsilon(t-2)$$

$$2: \quad X(t) - 0.90X(t-1) = \epsilon(t) + 0.80\epsilon(t-4).$$

That is, Model 1 is an ARMA(2,2), while Model 2 is an ARMA(1,4) where the first three MA coefficients are zero. In each of these models, both the correlogram and the partial correlogram exhibit an exponential decay, thus ruling out a pure AR or pure MA model. See Example 2.7 for the macro that generated this figure.

2.5.6. Subset and Multiplicative Subset ARMA Processes

A natural extension of the ARMA model is the case where only a few of its coefficients are nonzero; that is, the subset ARMA models which we write as

$$\left(\sum_{k=0}^P \theta_k L^{u_k} \right) X(t) = \left(\sum_{m=0}^Q \gamma_m L^{v_m} \right) \epsilon(t),$$

where $u_1 \leq \dots \leq u_P = p$ and $v_1 \leq \dots \leq v_Q = q$ are called the AR and MA lags, respectively. We can go a step further and define a multiplicative subset ARMA process as a combination of a “full” ARMA and subset ARMA by

$$\left(\sum_{j=0}^p \alpha_j L^j \right) \left(\sum_{k=0}^P \theta_k L^{u_k} \right) X(t) = \left(\sum_{l=0}^q \beta_l L^l \right) \left(\sum_{m=0}^Q \gamma_m L^{v_m} \right) \epsilon(t).$$

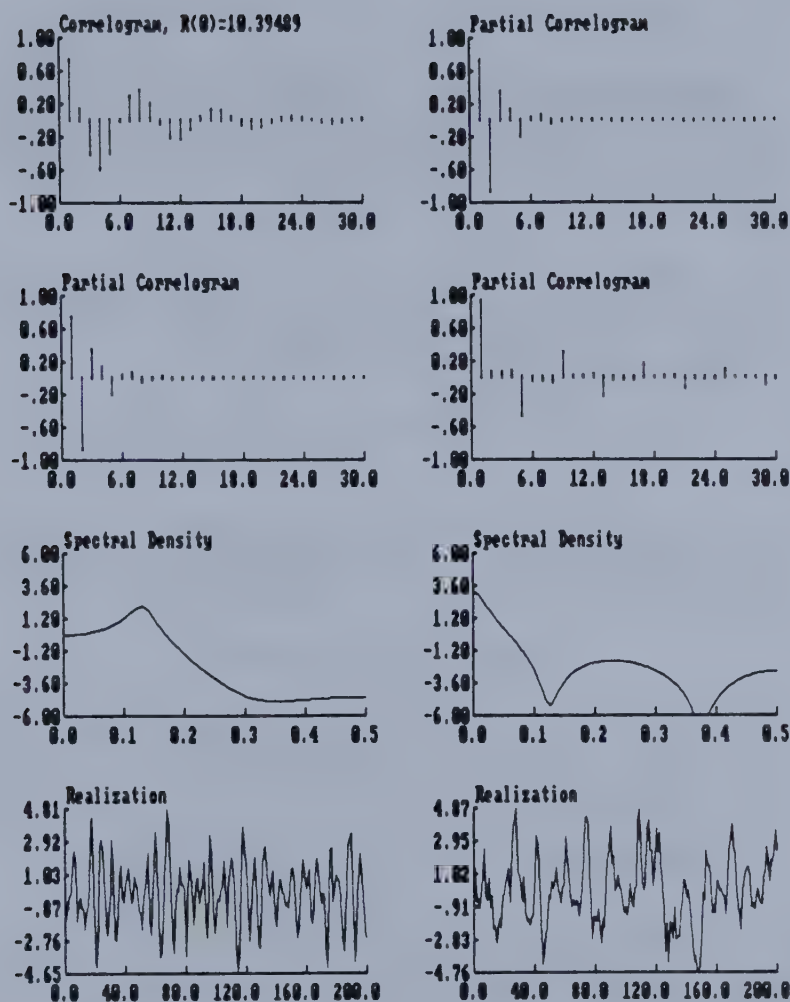


Figure 2.8. Correlogram, Partial Correlogram, Spectral Density, and a Realization of Length 200 from Two ARMA Processes.

Such models will be important in Chapter 3. Note that as long as the zeros of each of the four polynomials involved in the model are all outside the unit circle, then we can express the multiplicative subset ARMA model as an ordinary ARMA model of orders $p + u_P$ and $q + v_Q$, and use all of the results that we have obtained for ARMA processes.

2.5.7. ARIMA Processes

Many of the processes in economics and business are nonstationary but can be transformed to stationarity by differencing. This fact has led to the popularity of the so-called Box-Jenkins method of time series analysis (see Box and Jenkins (1970)). The method employs extensively what is called an ARIMA model.

Definition. A time series X is called an autoregressive integrated moving average process of orders (p, d, q) if the series $Z(t) = (1 - L)^d X(t)$ is an ARMA(p, q) process. We denote such a process by

$$X \sim \text{ARIMA}(p, d, q, \alpha, \beta, \sigma^2).$$

The simplest example of such a model is the random walk process where $d = 1$ and $p = q = 0$. Note that if we let g and h denote the AR and MA operators for the ARMA part of the model, then we have that X satisfies

$$g(L)(1 - L)^d X(t) = h(L)\epsilon(t).$$

The difficulty with ARIMA processes is in expressing exactly what is meant by this equation. As we have discussed for the random walk model, there is no MA(∞) representation for X . However, we can visualize specifying some initial conditions and then generating all of the rest of the X 's by the difference equation.

As a simple example, let $p = 1$, $q = 1$, and $d = 2$. This gives

$$(1 + \alpha L)(1 - L)^2 X(t) = (1 + \beta L)\epsilon(t),$$

that is,

$$X(t) + (\alpha - 2)X(t - 1) + (1 - 2\alpha)X(t - 2) + \alpha X(t - 3) = \epsilon(t) + \beta\epsilon(t - 1).$$

If we specify any three consecutive X 's, we can then construct an infinite series of future X 's in terms of a WN(σ^2) process ϵ by this difference equation which can also be solved to obtain an expression for past X 's. Note that the variance of these X 's grows without bound as t increases as in the random walk case. This sort of specification of initial conditions is implicit in any definition of an ARIMA model and should be kept in mind. For example, the joint pdf of $X(1), \dots, X(n)$ for an ARIMA model depends on what initial conditions one is assuming (see Ansley and Kohn (1985)).

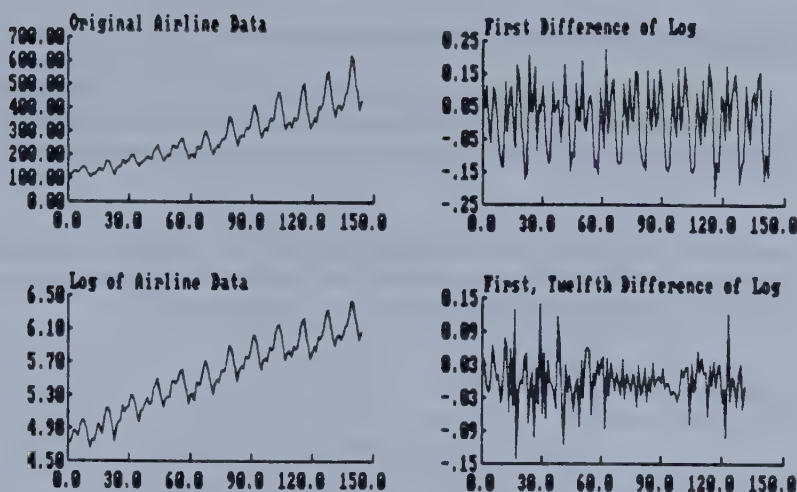


Figure 2.9. The Airline Data and Successive Transformations.

Multiplicative Subset Seasonal ARIMA Models

A general model of the form

$$g(L)G(L)(1-L)^d(1-L^S)^D X(t) = h(L)H(L)\epsilon(t),$$

where g , G , h , and H are the four operators in the multiplicative subset ARMA model, provides a general framework within which many seasonal time series can be analyzed. If the AR and MA lags are multiples of the seasonality factor S , then this model is the one which is used in the Box-Jenkins forecasting method (see Section 3.6). The most well known example of this kind of model is the international airline passengers data (Series VI in Section 1.1). In Figure 2.9 we give the original data and then the results of successively taking natural logs, first differences of the logs, and then 12th differences of the first difference of the log. Note how the log transform seems to stabilize the variance, the first difference removes the linear trend, and the 12th difference removes the seasonal cycle. Box and Jenkins suggest that a model with $p = 0$, $P = 0$, $q = 1$, and $Q = 1$ with $v_1 = 12$ fits the first and 12th differences of the log of the airline data.

2.5.8. ARMA Models and Inverse Autocovariances

Let X be a covariance stationary time series with autocovariance function R and spectral density f where f is bounded above and below by positive, finite constants. Then we can make the following definition (see Cleveland (1972)).

Definition. The inverse spectral density function fi , inverse autocovariance function Ri , and inverse autocorrelation function ρi of a covariance stationary time series having spectral density function f are given by

$$fi(\omega) = \frac{1}{f(\omega)}, \quad \omega \in [0, 1]$$

$$Ri(v) = \int_0^1 fi(\omega) e^{2\pi i v \omega} d\omega, \quad v \in \mathcal{Z}$$

$$\rho i(v) = \frac{Ri(v)}{Ri(0)}, \quad v \in \mathcal{Z}.$$

These functions are useful for ARMA models as indicated by the following theorem.

Theorem 2.5.7	INVERSE PARAMETERS FOR ARMA MODELS
----------------------	---

Let $X \sim \text{AR}(p, \alpha, \sigma^2)$, Y be an invertible $\text{MA}(q, \beta, \sigma^2)$, and Z be an invertible $\text{ARMA}(p, q, \alpha, \beta, \sigma^2)$. Then

a) The inverse spectral density fi_X and autocovariance function Ri_X of X are given by

$$\begin{aligned} fi_X(\omega) &= \frac{1}{\sigma^2} |g(e^{2\pi i \omega})|^2 \\ &= \sum_{v=-p}^p Ri_X(v) \cos 2\pi v \omega \\ &= Ri_X(0) + 2 \sum_{v=1}^p Ri_X(v) \cos 2\pi v \omega, \end{aligned}$$

where

$$Ri_X(v) = \begin{cases} \frac{1}{\sigma^2} \sum_{j=0}^{p-|v|} \alpha_j \alpha_{j+|v|}, & |v| \leq p \\ 0, & |v| > p; \end{cases}$$

that is, f_{iX} and R_{iX} are the spectral density and autocovariance functions of an $MA(p, \alpha, 1/\sigma^2)$ process.

b) The inverse spectral density f_{iY} and inverse autocovariance function R_{iY} of Y satisfy

$$f_{iY}(\omega) = \frac{1}{\sigma^2} \frac{1}{|h(e^{2\pi i \omega})|^2}$$

$$\sum_{k=0}^q \beta_k R_{iY}(k-v) = \delta_v \frac{1}{\sigma^2}, \quad v \geq 0;$$

that is, f_{iY} and R_{iY} are the spectral density and autocovariance functions of an $AR(q, \beta, 1/\sigma^2)$ process.

c) f_{iZ} and R_{iZ} are the spectral density and autocovariance functions of an $ARMA(q, p, \beta, \alpha, 1/\sigma^2)$ process.

We will use the duality between AR and MA processes described by this theorem to aid in model identification in Section 3.4.

2.5.9. The General Linear Process

In Chapter 3 we will describe the statistical properties of estimators of R and f for a time series X under as general a set of assumptions about X as possible. One popular such general assumption about a time series is that it is a general linear process.

Definition. The time series X is said to be a general linear process with coefficients β and errors ϵ if

$$X(t) = \mu + \sum_{j=-\infty}^{\infty} \beta_j \epsilon(t-j), \quad t \in \mathbb{Z},$$

where $\epsilon \sim WN(\sigma^2)$ and the β 's are square summable, that is, $\sum_{j=-\infty}^{\infty} \beta_j^2 < \infty$.

Note that the square summability of the β 's means that $X(t)$ exists as a limit in mean square. The following theorem presents some properties of the general linear process that will be useful in Chapter 3.

Theorem 2.5.8

PROPERTIES OF THE GENERAL LINEAR PROCESS

If X is a general linear process, then

a) $R(v) = \sigma^2 \sum_{j=-\infty}^{\infty} \beta_j \beta_{j+|v|}, \quad v \in \mathcal{Z}.$

b) R is absolutely summable and thus

$$f(\omega) = \sigma^2 \left| \sum_{j=-\infty}^{\infty} \beta_j e^{2\pi i j \omega} \right|^2, \quad \omega \in [0, 1].$$

Note that part (b) follows from the fact that a convolution (such as $R(v)$) of two sequences is absolutely summable if the individual sequences (in this case the sequence of β 's with itself) are absolutely summable (see Fuller (1976), p. 28). It is also easy to see that ARMA processes are general linear processes.

2.5.10. Nonlinear and Non-Gaussian Processes

In this book we are concerned primarily with processes that can be expressed in the linear form described in Section 2.5.9. A wide variety of authors are currently investigating the analysis of processes that are not linear and/or not Gaussian. A simple example of such a process is given in Problem C2.10. The interested reader can consult Section 11.5 of Priestley (1981) for a general introduction to nonlinear processes.

2.6. Algorithms for Univariate Time Series

In this section we discuss some of the algorithms that are used in univariate time series analysis. Thus the reader may want to give this section a cursory reading to get an overall view of the algorithms and how they are used in the TIMESLAB commands unless particularly interested in the details of the computations in TIMESLAB.

We begin by considering methods for solving prediction normal equations. These algorithms yield a host of information useful in performing other tasks as well. While there is a vast literature on prediction algorithms, most attention has been paid by statisticians to the use of either Cholesky factorization or the Kalman Filter Algorithm. While both of these procedures are calculating the same quantities, we will study both as each makes clear what happens in prediction in a different way.

2.6.1. Prediction Using the Cholesky Decomposition

We can use the modified Cholesky decomposition (MCD, see Section A.1.1) to obtain a series of results about calculating finite memory predictors. For the first, we follow Newton and Pagano (1982).

Theorem 2.6.1

PREDICTION VIA THE MCD

Let X be a purely nondeterministic covariance stationary time series with autocovariance function R and let $\gamma_0 = 1, \gamma_1, \dots$ be the coefficients of the $MA(\infty)$ representation of X . For $K = 1, 2, \dots$, let

$$\mathbf{\Gamma}_K = \text{Toepl}(R(0), \dots, R(K-1))$$

be the covariance matrix of $\mathbf{X}_K = (X(1), \dots, X(K))^T$, and $\mathbf{\Gamma}_K = \mathbf{L}_K \mathbf{D}_K \mathbf{L}_K^T$ be the MCD of $\mathbf{\Gamma}_K$. Define $\mathbf{e}_K = (e(1), \dots, e(K))^T$ as the solution to $\mathbf{L}_K \mathbf{e}_K = \mathbf{X}_K$ and note that the vectors $\mathbf{e}_1, \mathbf{e}_2, \dots$ are also nested. Then the finite memory best linear unbiased predictors and prediction error variances are given by, for $h \geq 1$,

a) $\hat{X}_{nh} = \sum_{k=h}^{n+h-1} L_{n+h, n+h-k} e(n+h-k).$

b) $\hat{\sigma}_{nh}^2 = \sum_{k=0}^{h-1} L_{n+h, n+h-k}^2 D_{n+h-k, n+h-k}.$

c) Further,

$$\lim_{n \rightarrow \infty} L_{n, n-j} = \gamma_j, \quad j \geq 1$$

$$\lim_{n \rightarrow \infty} D_{nn} = \sigma_{\infty 1}^2.$$

Part (c) gives a method for finding the γ 's if we know the function R . If we compare parts (a) and (b) to the corresponding formulas for infinite memory prediction in part (d) of Theorem 2.4.1, we see how the elements of the MCD are playing the role for finite memory predictors that the γ 's do for infinite memory prediction. This theorem also gives us one method of arriving at what is called the prediction error variance representation of the Gaussian likelihood of the parameters of time series. This likelihood is actually the joint pdf of a finite realization from a series.

Theorem 2.6.2

GAUSSIAN LIKELIHOOD FUNCTION

Under the conditions and notation of Theorem 2.6.1., we have

$$a) e(j) = \begin{cases} X(1), & j = 1 \\ X(j) - \hat{X}_{j-1,1}, & j > 1. \end{cases}$$

$$b) D_{jj} = \begin{cases} R(0), & j = 1 \\ \hat{\sigma}_{j-1,1}^2, & j > 1. \end{cases}$$

c) If X is Gaussian, then the joint pdf of \mathbf{X}_n , also called the likelihood function, is given by

$$f(\mathbf{x}) = (2\pi)^{-n/2} \left(\prod_{j=1}^n D_{jj} \right)^{-1/2} \exp \left(-\frac{1}{2} \sum_{j=1}^n \frac{e^2(j)}{D_{jj}} \right),$$

that is, the pdf is only a function of $X(1)$, $R(0)$, and the one step ahead prediction errors e and error variances D_{jj} for $j = 2, \dots, n$.

Proof: To prove part (a), note that $e(1) = \mathbf{L}_1^{-1} \mathbf{X}_1 = X(1)$, while for $j > 1$, we have that $X(j)$ is the j th row of \mathbf{L}_j times the vector \mathbf{e}_j ; that is,

$$\begin{aligned} X(j) &= \sum_{l=1}^j L_{jl} e(l) \\ &= e(j) + \sum_{l=1}^{j-1} L_{jl} e(l) \\ &= e(j) + \hat{X}_{j-1,1}, \end{aligned}$$

since $L_{jj} = 1$ and by part (a) of Theorem 2.6.1. For part (b), we have $\mathbf{L}_1 \mathbf{D}_1 \mathbf{L}_1^T = D_{11} = \Gamma_1 = R(0)$, while for $j > 1$, $D_{jj} = \hat{\sigma}_{j-1,1}^2$ by part (b) of Theorem 2.6.1. For part (c), we have (see Section A.4.2)

$$f(\mathbf{x}) = (2\pi)^{-n/2} |\Gamma_n|^{-1/2} \exp -\frac{1}{2} \mathbf{x}^T \Gamma_n^{-1} \mathbf{x}.$$

But

$$|\Gamma_n| = |\mathbf{L}_n \mathbf{D}_n \mathbf{L}_n^T| = |\mathbf{L}_n| |\mathbf{D}_n| |\mathbf{L}_n^T| = |\mathbf{D}_n| = \prod_{j=1}^n D_{jj},$$

since the determinant of a triangular matrix is just the product of its diagonal elements (which in the case of \mathbf{L}_n are just ones). Further, $\mathbf{\Gamma}_n^{-1} = \mathbf{L}_n^{-T} \mathbf{D}_n^{-1} \mathbf{L}_n^{-1}$, and thus

$$\mathbf{x}^T \mathbf{\Gamma}_n^{-1} \mathbf{x} = \mathbf{x}^T \mathbf{L}_n^{-T} \mathbf{D}_n^{-1} \mathbf{L}_n^{-1} \mathbf{x} = \mathbf{z}^T \mathbf{D}_n^{-1} \mathbf{z},$$

where $\mathbf{z} = \mathbf{L}_n^{-1} \mathbf{x} = \mathbf{e}_n$.

Part (c) is important for finding maximum likelihood estimators for the parameters of ARMA processes. We will see that the Kalman Filter Algorithm is ideally suited for calculating the ϵ 's and D 's.

The Levinson Recursion

In the previous two theorems we found finite memory linear predictors and prediction error variances without actually calculating the prediction coefficients λ_{nh} . When $h = 1$ there is a well known algorithm due to Levinson (1947) for recursively calculating these coefficients for memory $1, 2, \dots$. This algorithm is based on the Toeplitz form of the covariance matrix of a covariance stationary time series (see Friedlander et al. (1979) for a general description of algorithms for Toeplitz and near-Toeplitz matrices), and as we shall see is very important in a number of contexts. In the following theorem we omit the second subscript on λ_{nh} and $\hat{\sigma}_{nh}^2$ since $h = 1$ throughout.

Theorem 2.6.3 THE LEVINSON RECURSION

Let X be a purely nondeterministic time series with autocovariance function R and let

$$\mathbf{\Gamma}_n = \text{Toepl}(R(0), \dots, R(n-1)) = \mathbf{L}_n \mathbf{D}_n \mathbf{L}_n^T$$

be the MCD of the covariance matrix of $\mathbf{X}_n = (X(1), \dots, X(n))^T$. Let $\lambda_1, \lambda_2, \dots$ be the prediction coefficients of $\hat{X}_{11}, \hat{X}_{21}, \dots$, and let

$$\hat{\sigma}_j^2 = \text{Var}(X(j+1) - \hat{X}_{j1}),$$

with $\hat{\sigma}_0^2 = R(0)$. Then

a) For $n \geq 1$,

$$\mathbf{L}_n^{-T} = \mathbf{U}_n = \begin{bmatrix} 1 & -\lambda_1(1) & -\lambda_2(2) & \cdots & -\lambda_{n-1}(n-1) \\ & 1 & -\lambda_2(1) & \cdots & -\lambda_{n-1}(n-2) \\ & & 1 & \cdots & -\lambda_{n-1}(n-3) \\ & & & \ddots & \vdots \\ & & & & -\lambda_{n-1}(1) \\ & & & & & 1 \end{bmatrix},$$

that is, the (j, k) th element of \mathbf{U}_n is 0 for $j > k$, 1 for $j = k$, and $-\lambda_{k-1}(k-j)$ for $j < k$. Further,

$$\mathbf{D}_n^{-1} = \mathbf{V}_n = \text{Diag}\left(\frac{1}{\hat{\sigma}_0^2}, \frac{1}{\hat{\sigma}_1^2}, \dots, \frac{1}{\hat{\sigma}_{n-1}^2}\right).$$

b) The λ 's and $\hat{\sigma}^2$'s satisfy the following recursion for $j \geq 1$:

$$\lambda_{j+1}(j+1) = \frac{R(j+1) - \sum_{k=1}^j \lambda_j(k) R(j+1-k)}{\hat{\sigma}_j^2}$$

$$\lambda_{j+1}(k) = \lambda_j(k) - \lambda_{j+1}(j+1) \lambda_j(j+1-k), \quad k = 1, \dots, j$$

$$\hat{\sigma}_{j+1}^2 = \hat{\sigma}_j^2 (1 - \lambda_{j+1}^2(j+1)),$$

with $\lambda_1(1) = \rho(1)$, $\hat{\sigma}_0^2 = R(0)$, and $\hat{\sigma}_1^2 = \hat{\sigma}_0^2 (1 - \lambda_1^2(1))$.

Proof: To show part (a) we need only show that $\mathbf{\Gamma}_n^{-1} = \mathbf{U}_n \mathbf{V}_n \mathbf{U}_n^T$ or equivalently that $\mathbf{\Gamma}_n \mathbf{U}_n \mathbf{V}_n \mathbf{U}_n^T = \mathbf{I}_n$, since then $\mathbf{\Gamma}_n = \mathbf{U}_n^{-T} \mathbf{V}_n^{-1} \mathbf{U}_n^{-1}$, and since \mathbf{U}_n^{-T} is unit lower triangular, \mathbf{V}_n^{-1} is diagonal with positive diagonal elements, and the MCD is unique, we must have $\mathbf{L}_n = \mathbf{U}_n^{-1}$ and $\mathbf{D}_n = \mathbf{V}_n^{-1}$. We show $\mathbf{\Gamma}_n \mathbf{U}_n \mathbf{V}_n \mathbf{U}_n^T = \mathbf{I}_n$ by induction. The claim is clearly true when $n = 1$. For $k \geq 1$, define $\mathbf{r}_k = (R(1), \dots, R(k))^T$. Then

$$\begin{aligned} \mathbf{\Gamma}_{k+1} \mathbf{U}_{k+1} &= \begin{bmatrix} \mathbf{\Gamma}_k & \mathbf{P}_k \mathbf{r}_k \\ \mathbf{r}_k^T \mathbf{P}_k & R(0) \end{bmatrix} \begin{bmatrix} \mathbf{U}_k & -\mathbf{P}_k \boldsymbol{\lambda}_k \\ \mathbf{0}_k^T & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{\Gamma}_k \mathbf{U}_k & \mathbf{P}_k \mathbf{r}_k - \mathbf{\Gamma}_k \mathbf{P}_k \boldsymbol{\lambda}_k \\ \mathbf{r}_k^T \mathbf{P}_k \mathbf{U}_k & R(0) - \mathbf{r}_k^T \mathbf{P}_k^2 \boldsymbol{\lambda}_k \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{\Gamma}_k \mathbf{U}_k & \mathbf{0}_k \\ \mathbf{r}_k^T \mathbf{P}_k \mathbf{U}_k & \hat{\sigma}_k^2 \end{bmatrix} \end{aligned}$$

by part (b) of Theorem 2.4.1 and the properties of the permutation matrix. Thus

$$\mathbf{\Gamma}_{k+1} \mathbf{U}_{k+1} \mathbf{V}_{k+1} \mathbf{U}_{k+1}^T = \begin{bmatrix} \mathbf{\Gamma}_k \mathbf{U}_k & \mathbf{0}_k \\ \mathbf{r}_k^T \mathbf{P}_k \mathbf{U}_k & \hat{\sigma}_k^2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_k \mathbf{U}_k^T & \mathbf{0}_k \\ -\frac{1}{\hat{\sigma}_k^2} \boldsymbol{\lambda}_k^T \mathbf{P}_k & \frac{1}{\hat{\sigma}_k^2} \end{bmatrix} = \mathbf{I}_{k+1}$$

by the induction hypothesis and the fact that

$$\mathbf{r}_k^T \mathbf{P}_k \mathbf{U}_k \mathbf{V}_k \mathbf{U}_k^T = \mathbf{r}_k^T \mathbf{P}_k \mathbf{\Gamma}_k^{-1} = \mathbf{r}_k^T \mathbf{\Gamma}_k^{-1} \mathbf{P}_k = \boldsymbol{\lambda}_k^T \mathbf{P}_k,$$

since $\mathbf{P}_k \mathbf{\Gamma}_k^{-1} \mathbf{P}_k = \mathbf{\Gamma}_k^{-1}$ and by part (b) of Theorem 2.4.1. To prove part (b) we now use part (a) and more algebra to show that

$$\begin{aligned} \lambda_{k+1} &= \mathbf{\Gamma}_{k+1}^{-1} \mathbf{r}_{k+1} \\ &= \begin{pmatrix} \lambda_k - \mathbf{P}_k \lambda_k \left[\frac{1}{\hat{\sigma}_k^2} (R(k+1) - \lambda_k^T \mathbf{P}_k \mathbf{r}_k) \right] \\ \frac{1}{\hat{\sigma}_k^2} (R(k+1) - \lambda_k^T \mathbf{P}_k \mathbf{r}_k) \end{pmatrix}, \end{aligned}$$

which is part (b).

Partial Autocorrelations

The coefficients $\lambda_1(1), \lambda_2(2), \dots$ play an important role in time series analysis. The next theorem shows that they are in fact partial correlation coefficients. The proof of the theorem is a straightforward application of the results of Theorem 2.6.3.

Theorem 2.6.4 PARTIAL AUTOCORRELATIONS

Let X be a purely nondeterministic time series and let \mathbf{X}_m be a realization of length m from X . Let $\hat{X}_{m+1|m}$ and $\hat{X}_{0|m}$ denote the best linear predictors of $X(m+1)$ and $X(0)$, respectively, given \mathbf{X}_m . Let

$$e_{m+1|m} = X(m+1) - \hat{X}_{m+1|m} \quad \text{and} \quad e_{0|m} = X(0) - \hat{X}_{0|m}.$$

Then

$$\text{a) } \hat{X}_{m+1|m} = \lambda_m^T \mathbf{P}_m \mathbf{X}_m, \quad \hat{X}_{0|m} = \lambda_m^T \mathbf{X}_m.$$

$$\text{b) } \text{Var}(e_{m+1|m}) = \text{Var}(e_{0|m}) = \hat{\sigma}_{m+1}^2.$$

c) $\lambda_{m+1}(m+1) = \text{Corr}(e_{m+1|m}, e_{0|m})$; that is, $\lambda_{m+1}(m+1)$ is the correlation between what is left in $X(m+1)$ and $X(0)$ after having removed their linear relationship with the X 's in between, that is, with \mathbf{X}_m . Thus $\lambda_m(m)$ is called the partial autocorrelation coefficient of lag m .

Note that

$$\hat{X}_{m+1|m} = \lambda_m(1)X(m) + \dots + \lambda_m(m)X(1)$$

$$\hat{X}_{0|m} = \lambda_m(m)X(m) + \dots + \lambda_m(1)X(1);$$

that is, the forward prediction coefficients are just the backward prediction coefficients in reverse order, while the forward and backward prediction error variances are the same. It is important to note that all of these results are based on the fact that the matrix Γ_n in the prediction normal equations is Toeplitz. We will see sample analogs of these results in Section 3.4.4.

Levinson's Recursions and AR Processes

In Section 2.5.4 we studied the autoregressive process of order p

$$X(t) + \alpha_1 X(t-1) + \cdots + \alpha_p X(t-p) = \epsilon(t),$$

where $\epsilon \sim \text{WN}(\sigma^2)$ and the zeros of $g(z) = \sum_{j=0}^p \alpha_j z^j$ are all greater than one in modulus. We showed that $\alpha = (\alpha_1, \dots, \alpha_p)^T$ satisfies the Yule-Walker equations

$$\Gamma_p \alpha = -\mathbf{r}_p.$$

Thus we have that $\alpha = -\lambda_p$ and we can use Levinson's algorithm to calculate α . If we let $\alpha_j = (\alpha_j(1), \dots, \alpha_j(j))^T = -\lambda_j$, we can rewrite part (b) of Theorem 2.6.3 as (omitting the hats on all of the quantities involved):

$$\begin{aligned} \alpha_{j+1}(j+1) &= -\frac{R(j+1) - \sum_{k=1}^j \alpha_j(k) R(j+1-k)}{\sigma_j^2} \\ \alpha_{j+1}(k) &= \alpha_j(k) + \alpha_{j+1}(j+1) \alpha_j(j+1-k), \quad k = 1, \dots, j \\ \sigma_{j+1}^2 &= \sigma_j^2 (1 - \alpha_{j+1}^2(j+1)), \end{aligned}$$

with $\alpha_1(1) = -\rho(1)$ and $\sigma_0^2 = R(0)$, and note that $\alpha = \alpha_p$ and $\sigma_p^2 = \sigma^2$.

It is also easy to see that for $m > p$,

$$\alpha_m = \begin{pmatrix} \alpha \\ \mathbf{0}_{m-p} \end{pmatrix},$$

and thus the structure of the MCD of Γ_n for an AR(p) process is very simple.

Theorem 2.6.5 THE MCD OF Γ_p FOR AN AR(p)

Let

$$X \sim \text{AR}(p, \alpha, \sigma^2),$$

and for $j \geq 1$, let α_j and σ_j^2 denote the solutions to

$$\begin{aligned} \Gamma_j \alpha_j &= -\mathbf{r}_j \\ \sigma_j^2 &= R(0) - \mathbf{r}_j^T \Gamma_j^{-1} \mathbf{r}_j. \end{aligned}$$

Let $\Gamma_n = L_n D_n L_n^T$ be the MCD of the covariance matrix of

$$X_n = (X(1), \dots, X(n))^T.$$

Then

a) For $m > p$,

$$\alpha_m = \begin{pmatrix} \alpha \\ 0_{m-p} \end{pmatrix},$$

and $\sigma_m^2 = \sigma^2$, $\alpha_m(m) = -\lambda_m(m) = 0$.

b) For $j > p + 1$, the j th row of L_n^{-1} is given by

$$(0_{j-p-1}^T, \alpha_p, \dots, \alpha_1, 1, 0_{n-j}^T),$$

while the upper left-hand $(p+1 \times p+1)$ corner of L_n^{-1} is the matrix

$$A = \begin{bmatrix} 1 & & & & & & \\ \alpha_1(1) & 1 & & & & & \\ \alpha_2(2) & \alpha_2(1) & 1 & & & & \\ \vdots & \vdots & \ddots & \ddots & & & \\ \alpha_{p-1}(p-1) & \alpha_{p-1}(p-2) & \dots & \alpha_{p-1}(1) & 1 & & \\ \alpha_p & \alpha_{p-1} & \dots & \alpha_2 & \alpha_1 & 1 & \end{bmatrix}.$$

Thus there are only $p+(p-1)p/2$ distinct elements in L_n^{-1} . We will use this fact to simulate data from a Gaussian AR process and when doing regression analysis for a model having AR errors.

Testing AR Processes for Stationarity

Given coefficients $\alpha_1, \dots, \alpha_p$, we would like a simple means for testing whether the zeros of $g(z) = \sum_{j=0}^p \alpha_j z^j$ are all outside the unit circle. One obvious test is to actually find the zeros (see the TIMESLAB command POLY-ROOTS) but there is another simpler test which is contained in the next theorem.

Theorem 2.6.6 THE SCHUR TEST FOR STATIONARITY

Let $\alpha_1, \dots, \alpha_p$ be a set of constants, and let U and V be the $(p \times p)$ lower triangular Toeplitz matrices having first columns $(1, \alpha_1, \dots, \alpha_{p-1})^T$ and $(\alpha_p, \dots, \alpha_1)^T$, respectively. Then

a) The zeros of $g(z) = \sum_{j=0}^p$ are all greater than one in modulus if and only if the $(p \times p)$ matrix

$$\mathbf{S} = \mathbf{U}\mathbf{U}^T - \mathbf{V}\mathbf{V}^T$$

is positive definite.

b) If \mathbf{S} is positive definite, then

$$\mathbf{\Gamma}_p = \sigma^2 \mathbf{S}^{-1}$$

is the covariance matrix of an $AR(p, \boldsymbol{\alpha}, \sigma^2)$ process.

c) $\mathbf{\Gamma}_p$ is positive definite if and only if the partial autocorrelations of lags 1 through p are all less than one in absolute value.

Note that part (a) is due to Schur (see Pagano (1973)), and thus we will refer to the matrix \mathbf{S} as the Schur matrix of $\boldsymbol{\alpha}$. The results of this theorem suggest a simple test for whether the zeros of g are all outside the unit circle. This check is incorporated in the **ARPART** command.

The ARPART Command

Given a set of coefficients $\alpha_1, \dots, \alpha_p$, we can test the zeros of g and obtain the partial autocorrelations as a byproduct by essentially running Levinson's algorithm backward. For $1 \leq k \leq j$, we have

$$\lambda_j(k) = \lambda_{j+1}(k) + \lambda_{j+1}(j+1)\lambda_j(j+1-k)$$

$$\lambda_j(j+1-k) = \lambda_{j+1}(j+1-k) + \lambda_{j+1}(j+1)\lambda_j(k),$$

which gives

$$\lambda_j(k) = \frac{\lambda_{j+1}(k) + \lambda_{j+1}(j+1)\lambda_{j+1}(j+1-k)}{1 - \lambda_{j+1}^2(j+1)}, \quad k = 1, \dots, j.$$

The algorithm starts with $\lambda_p(k) = -\alpha_k$ and proceeds for $j = p-1, \dots, 1$. If any of the partials are greater than one in absolute value, then we know that the zeros of g are not all outside the unit circle.

In the command

```
part=ARPART(alpha,p,ier)
```

the output integer variable **ier** is zero if none of the partials are greater than one in absolute value, and is equal to j if $\lambda_j(j)$ is the first value (starting with $j = p$) greater than one in absolute value.

The PARTAR Command

We can also easily derive an algorithm for finding the coefficients α of an $\text{AR}(p)$ process corresponding to a set of partial autocorrelations. Thus given $\lambda_1(1), \dots, \lambda_p(p)$, we can find recursively

$$\lambda_{j+1}(k) = \lambda_j(k) + \lambda_{j+1}(j+1)\lambda_j(j+1-k), \quad k = 1, \dots, j,$$

for $j = 1, \dots, p-1$, and then $\alpha_j = -\lambda_p(j)$, for $j = 1, \dots, p$. The command

`alpha=PARTAR(part,p)`

carries out this algorithm. Note that `PARTAR` does not check whether or not the absolute values of `part` are less than one.

The ARCORR Command

The results of the last few theorems also suggest a method for finding the autocorrelations of an $\text{AR}(p, \alpha, \sigma^2)$ process given its parameters α and σ^2 . First we have

$$\begin{aligned} \sigma^2 &= \sigma_p^2 = \sigma_{p-1}^2(1 - \alpha_p^2(p)) \\ &= \sigma_{p-2}^2(1 - \alpha_{p-1}^2(p-1))(1 - \alpha_p^2(p)) \\ &= \dots \\ &= \sigma_0^2 \prod_{j=1}^p (1 - \alpha_j^2(j)). \end{aligned}$$

But $\sigma_0^2 = R(0)$ and so

$$R(0) = \frac{\sigma^2}{\prod_{j=1}^p (1 - \alpha_j^2(j))}.$$

Next we have that α_j satisfies $\mathbf{\Gamma}_j \alpha_j = -\mathbf{r}_j$, the last row of which gives

$$R(j) = - \sum_{i=1}^j \alpha_j(i) R(j-i), \quad j \geq 1.$$

Then $\rho(v) = R(v)/R(0)$. Thus the algorithm consists of (1) running Levinson's algorithm backward as in `ARPART` until $R(0)$ is determined, (2) running Levinson's algorithm forward for $j = 1, \dots, p$, at each order j finding $R(j)$, and

finally (3) if the number **M** of desired correlations is greater than p , using the Yule-Walker equation

$$R(p+k) = - \sum_{j=1}^p \alpha_j R(p+k-j), \quad k \geq 1,$$

to find the remaining R 's. Again in the command

```
rho=ARCORR(alpha,p,rvar,M,R0,ier)
```

ier indicates whether the zeros of g are all greater than one in absolute value in the same way as in **ARPART**. Note that **M** is the number of autocorrelations that are desired.

The CORRAR Command

The command

```
alpha=CORRAR(rho,R0,p,rvar)
```

uses Levinson's algorithm to find AR parameters **alpha** and **rvar** from correlations **rho** and variance **R0**.

The ARDT Command

To simulate a Gaussian $AR(p)$ process, **ARDT** generates the initial values \mathbf{X}_p to have covariance matrix $\mathbf{\Gamma}_p$ and then uses the AR difference equation to obtain the rest of the observations. Thus **ARDT** does Levinson's algorithm backward to obtain \mathbf{L}_p and \mathbf{D}_p and then solves the triangular system of equations (see Theorem 2.6.5)

$$\mathbf{D}_p^{-1/2} \mathbf{L}_p^{-1} \mathbf{X}_p = \mathbf{e}_p,$$

where $\mathbf{e}_p \sim N_p(\mathbf{0}_p, \mathbf{I}_p)$, that is,

$$X(j) = \begin{cases} e(1)\sqrt{D_{11}}, & j = 1 \\ e(j)\sqrt{D_{jj}} - \sum_{k=1}^{j-1} \alpha_{j-1}(k)X(j-k), & j = 2, \dots, p \\ e(j)\sqrt{\sigma^2} - \sum_{k=1}^p \alpha(k)X(j-k), & j = p+1, \dots, n. \end{cases}$$

This is done in the form

```
x=ARDT(alpha,p,rvar,seed,n,ier,R0)
```

which returns the stationarity check **ier** and the variance of the process **R0** as a byproduct. If the process is not stationary, then **ARDT** will abort and not

generate a realization. The ARDT command can also be used to generate values of any difference equation of the form

$$X(t) + \alpha_1 X(t-1) + \cdots + \alpha_p X(t-p) = \epsilon(t)$$

for any user-supplied $X(1), \dots, X(p)$ and $\epsilon(p+1), \dots, \epsilon(n)$. The first p X 's and $\epsilon(p+1), \dots, \epsilon(n)$ are entered in the single array \mathbf{f} in the command

$$\mathbf{x} = \text{ARDT}(\alpha, p, n, \mathbf{f})$$

and the initial values and generated values are returned in the array \mathbf{x} .

The ARFILT Command

In Section A.5.3 we discuss transforming a regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\text{Var}(\boldsymbol{\epsilon}) = \gamma^2 \mathbf{V}$, into one having uncorrelated errors by multiplying the observation vector \mathbf{y} and design matrix \mathbf{X} by $\mathbf{D}^{-1/2} \mathbf{L}^{-1}$, that is, by the inverse square root of \mathbf{V} . The ARFILT command does such a transformation when \mathbf{V} is the $(n \times n)$ covariance matrix of an $\text{AR}(p, \boldsymbol{\alpha}, \sigma^2)$ process. In Section 3.7.1 we give a macro for solving a regression problem that has autoregressive errors.

The MCD for ARMA Processes

We have seen that the Cholesky factors for an AR covariance matrix take on a very special form. For the ARMA case, the lower triangular matrix \mathbf{L}_n can be further decomposed into two parts, one corresponding to the AR part of the model and one corresponding to the MA part.

It would be nice if these two parts corresponded exactly to the triangular factors for an $\text{AR}(p, \boldsymbol{\alpha}, \sigma^2)$ and an $\text{MA}(q, \boldsymbol{\beta}, \sigma^2)$, respectively, but this is not possible. There are two basic approaches to this problem; the first is due to Ansley (1979) and the second to Newton and Pagano (1982). The next theorem describes the second approach. It essentially decomposes \mathbf{L}_n into the product of the \mathbf{L} for the AR part of the process and a matrix that is almost the \mathbf{L} for the MA part of the process.

Theorem 2.6.7 ARMA PREDICTION VIA THE MCD

Let $X \sim \text{ARMA}(p, q, \boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma^2)$ and $Z \sim \text{AR}(p, \boldsymbol{\alpha}, \sigma^2)$. Let

$$\boldsymbol{\Gamma}_{X,n} = \text{Toepl}(R_X(0), \dots, R_X(n-1)) = \mathbf{L}_{X,n} \mathbf{D}_{X,n} \mathbf{L}_{X,n}^T$$

and

$$\boldsymbol{\Gamma}_{Z,n} = \text{Toepl}(R_Z(0), \dots, R_Z(n-1)) = \mathbf{L}_{Z,n} \mathbf{D}_{Z,n} \mathbf{L}_{Z,n}^T$$

be the MCD of the covariance matrices of realizations \mathbf{X}_n and \mathbf{Z}_n of length n from X and Z , respectively. Define \mathbf{Y}_n and \mathbf{e}_n by

$$\mathbf{Y}_n = \mathbf{L}_{Z,n}^{-1} \mathbf{X}_n \quad \text{and} \quad \mathbf{e}_n = \mathbf{L}_{Y,n}^{-1} \mathbf{Y}_n,$$

where

$$\Gamma_{Y,n} = \text{Var}(\mathbf{Y}_n) = \mathbf{L}_{Z,n}^{-1} \Gamma_{X,n} \mathbf{L}_{Z,n}^{-T} = \mathbf{L}_{Y,n} \mathbf{D}_{Y,n} \mathbf{L}_{Y,n}^T.$$

Then

a) The sequences $\Gamma_{X,n}$, $\Gamma_{Y,n}$, $\Gamma_{Z,n}$, $\mathbf{L}_{X,n}$, $\mathbf{L}_{Y,n}$, $\mathbf{L}_{Z,n}$, $\mathbf{D}_{X,n}$, and $\mathbf{D}_{Y,n}$ of matrices and \mathbf{X}_n , \mathbf{Y}_n , and \mathbf{e}_n of vectors are all nested with respect to n , and thus for any n greater than or equal to both j and k , we can refer to the (j, k) th element of one of the matrices by dropping the subscript n and adding subscripts j and k . For example, $L_{Y,j,k}$ refers to the (j, k) th element of $\mathbf{L}_{Y,n}$ for any n greater than or equal to both j and k . Further, we can refer to the j th element of the vectors by $X(j)$, $Y(j)$, and $e(j)$ for any $n \geq j$.

$$\text{b) } \mathbf{L}_{X,n} = \mathbf{L}_{Z,n} \mathbf{L}_{Y,n} \quad \text{and} \quad \mathbf{D}_{X,n} = \mathbf{D}_{Y,n}.$$

c) For $j > p + q$, the only nonzero elements of the j th row of $\mathbf{L}_{Y,n}$ are $L_{Y,j,j-k}$ for $k = 0, \dots, q$. Further,

$$\lim_{j \rightarrow \infty} L_{Y,j,j-k} = \beta_k, \quad k = 0, \dots, q$$

$$\lim_{j \rightarrow \infty} D_{Y,j,j} = \sigma^2.$$

d) The memory n , h step ahead predictor \hat{X}_{nh} , and prediction error variance $\hat{\sigma}_{nh}^2$ are given by

$$\hat{X}_{nh} = \hat{Y}_{nh} - \sum_{j=1}^p \alpha_j \hat{X}_{n,h-j}$$

$$\hat{\sigma}_{nh}^2 = \sum_{k=0}^{h-1} L_{X,n+h,n+h-k}^2 D_{X,n+h-k,n+h-k},$$

where

$$\hat{Y}_{nh} = \begin{cases} \sum_{k=1}^q L_{Y,n+h,n+h-k} e(n+h-k), & h = 1, \dots, q \\ 0, & h > q \end{cases}$$

and $\hat{X}_{n,h-j} = X(n+h-j)$ if $j \geq h$.

e) If $q = 0$, that is, if $X \sim \text{AR}(p, \alpha, \sigma^2)$,

$$\hat{X}_{nh} = - \sum_{j=1}^p \alpha_j \hat{X}_{n, h-j}$$

$$\hat{\sigma}_{nh}^2 = \sum_{k=0}^{h-1} L_{Z, n+h, n+h-k}^2 D_{Z, n+h-k, n+h-k} = \sum_{k=0}^{h-1} \gamma_k^2, \quad \text{if } n > p,$$

where the γ 's are the coefficients of the $\text{MA}(\infty)$ representation of Z .

f) If $p = 0$, that is, if $X \sim \text{MA}(q, \beta, \sigma^2)$,

$$\hat{X}_{nh} = \begin{cases} \sum_{k=h}^q L_{X, n+h, n+h-k} e(n+h-k), & h = 0, \dots, q \\ 0, & h > q \end{cases}$$

$$\hat{\sigma}_{nh}^2 = \sum_{k=0}^{h-1} L_{X, n+h, n+h-k}^2 D_{X, n+h-k, n+h-k},$$

and $L_{X, n, n-k} \rightarrow \beta_k$, for $k = 0, \dots, q$, while $D_{X, n, n} \rightarrow \sigma^2$ as $n \rightarrow \infty$.

2.6.2. ARMA Prediction and the Kalman Filter Algorithm

Under some circumstances, the best linear unbiased predictors can be calculated very simply. An example is the Gaussian $\text{AR}(1)$ process

$$X(t) + \alpha X(t-1) = \epsilon(t),$$

since then X is a Markov process; that is, the conditional distribution of $X(n+1)$ given $X(1), \dots, X(n)$ is the same as the conditional distribution of $X(n+1)$ given just $X(n)$. This is intuitively clear and easily verified by finding the two distributions involved (see Problem T2.16).

Thus for a Gaussian $\text{AR}(1)$ we have

$$\begin{aligned} \tilde{X}_{n1} &= E[X(n+1)|X(1), \dots, X(n)] \\ &= E[X(n+1)|X(n)] \\ &= E[-\alpha X(n) + \epsilon(n+1)|X(n)] \\ &= -\alpha X(n), \end{aligned}$$

since $\epsilon(n+1)$ is independent of $X(n)$. Further,

$$\tilde{X}_{nh} = -\alpha \tilde{X}_{n, h-1} = \dots = (-\alpha)^h X(n).$$

Now if X satisfies a finite order difference equation, we will be able to construct a vector time series that is Markovian (see Problem T4.10) and use arguments similar to those above for an AR(1) to get predictors. The Kalman Filter Algorithm (see Section A.4.3) is ideal for doing prediction given a vector Markov process.

The Markovian Representation of an ARMA Process

There are several ways to determine a state and observational equation for an ARMA process. We describe the one given in Jones (1980).

Theorem 2.6.8 ARMA STATE AND OBSERVATION EQUATIONS

Let X be an $ARMA(p, q, \alpha, \beta, \sigma^2)$ and let

$$\mathbf{X}_t = \begin{bmatrix} X_{t,0} \\ X_{t,1} \\ \vdots \\ X_{t,r-1} \end{bmatrix},$$

where $r = \max(p, q + 1)$ and X_{th} is the infinite memory, h step ahead predictor of $X(t + h)$. Then

a) The \mathbf{X}_t satisfy the state and observational equations:

$$\mathbf{X}_t = \mathbf{A}\mathbf{X}_{t-1} + \mathbf{w}_t$$

$$X(t) = \mathbf{h}^T \mathbf{X}_t + v_t,$$

where \mathbf{h} is a vector of length r consisting of zeros except for its first element which is one, $\text{Var}(v_t) = 0$, and $\mathbf{w}_t = \mathbf{g}\epsilon(t)$ with $\mathbf{g} = (\gamma_0, \gamma_1, \dots, \gamma_{r-1})^T$, where the γ 's are the coefficients in the $MA(\infty)$ representation of X , while

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & 0 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -\alpha_r & -\alpha_{r-1} & -\alpha_{r-2} & \dots & -\alpha_2 & -\alpha_1 \end{bmatrix}$$

and $\alpha_k = 0$ if $k > p$.

b) $E(\mathbf{X}_0) = \mathbf{0}_r$, while the (j, k) th element of $\text{Var}(\mathbf{X}_0)$ is given by

$$\Sigma_{jk} = R(|j - k|) - \sigma^2 \sum_{l=0}^{\min(j-1, k-1)} \gamma_l \gamma_{l+|j-k|},$$

which is $R(0)$ if $j = k$.

Proof: From part (d) of Theorem 2.4.1 we have that the infinite memory, h step ahead predictor X_{th} of $X(t + h)$ and its error mean square σ_{th}^2 are given by

$$X_{th} = \sum_{k=h}^{\infty} \gamma_k \epsilon(t + h - k)$$

$$\sigma_{th}^2 = \sigma^2 \sum_{k=0}^{h-1} \gamma_k^2,$$

where the γ_k 's are the coefficients in the $\text{MA}(\infty)$ representation of X . Thus the first element of \mathbf{X}_t is just $X(t)$ which verifies that the observational equation above is valid. Note how simple this equation is. The sequence of matrices \mathbf{H}_t in the general formulation is the simple constant row vector \mathbf{h}^T , d is one, and the observational error covariance matrix \mathbf{V}_t is just a scalar zero.

We have for $j = 0, 1, \dots, r-1$ that

$$X_{t,j} = \gamma_j \epsilon(t) + \gamma_{j+1} \epsilon(t-1) + \gamma_{j+2} \epsilon(t-2) + \dots$$

$$X_{t-1,j+1} = \gamma_{j+1} \epsilon(t-1) + \gamma_{j+2} \epsilon(t-2) + \dots,$$

that is, $X_{t,j} = X_{t-1,j+1} + \gamma_j \epsilon(t)$. This gives a recursion for the elements of \mathbf{X}_t in terms of those in \mathbf{X}_{t-1} except for the last element of \mathbf{X}_t which is in terms of $X_{t-1,r}$ which is not an element of \mathbf{X}_{t-1} . But

$$\begin{aligned} X_{t-1,r} &= \mathbf{E}(X(t-1+r) | \epsilon(t-1), \epsilon(t-2), \dots) \\ &= - \sum_{j=1}^p \alpha_j \mathbf{E}(X(t-1+(r-j)) | \epsilon(t-1), \epsilon(t-2), \dots) \\ &\quad + \sum_{k=0}^q \beta_k \mathbf{E}(\epsilon(t-1+r-k) | \epsilon(t-1), \epsilon(t-2), \dots). \end{aligned}$$

Since $r = \max(p, q+1)$, we have $r > k$ in the second sum and thus each term in the sum is zero. Finally, the ϵ 's up to time $t-1$ are just a linear function of the X 's up to time $t-1$, so we have

$$\begin{aligned} X_{t-1,r-j} &= \mathbf{E}(X(t-1+(r-j)) | X(t-1), X(t-2), \dots) \\ &= \mathbf{E}(X(t-1+(r-j)) | \epsilon(t-1), \epsilon(t-2), \dots), \end{aligned}$$

and since $r \geq p$, we have $t-1+(r-j) > t-1$ which means each term in the first sum is an element of \mathbf{X}_{t-1} . Thus we have verified that the state equation is true.

It remains to verify that the starting values in part (b) are valid. The mean of \mathbf{X}_t is $\mathbf{0}_r$ since its elements are infinite memory predictors. The formula for the covariance matrix follows immediately from part (d,ii) of Theorem 2.4.1.

Implications: From the Kalman Filter Theorem (see Theorem A.4.4) we have that the first element of $\tilde{\mathbf{X}}_{t+h-1|t-1}$ and the (1,1) element of $\tilde{\Sigma}_{t+h-1|t-1}$ are the memory $t-1$, h step ahead predictor and prediction error variance of $X(t+h-1)$ given $X(1), \dots, X(t-1)$. To evaluate the Gaussian likelihood of an ARMA process we need only these terms for $h=1$ (see Theorem 2.6.2). Note that the transition matrix \mathbf{A} is the same for all t and has mostly zeros as elements.

The ARMAPRED Command

Given data $X(1), \dots, X(n)$ and the parameters of an ARMA process, ARMAPRED calculates memory t , h step ahead predictors and (optionally) prediction standard errors for $t = t_f, \dots, t_l$ and $h = h_f, \dots, h_l$. Note that $t_f \geq 1$, $t_l \leq n$, and $h_l \geq 1$.

2.6.3. The Cramer-Wold Factorization

An important problem in time series analysis is to find moving average parameters $\boldsymbol{\beta}$ and σ^2 corresponding to a given set of autocovariances $R(0), R(1), \dots, R(q)$. We will devote an unusual amount of attention to this problem because the algorithm usually used to compute it, and others that are very similar to it, are heavily used in a variety of ways in modern time series analysis.

The covariance generating function of X is

$$\Gamma(z) = \sum_{v=-q}^q R(v)z^v = \sigma^2(z)h(z)h(z^{-1}),$$

where $h(z) = \sum_{k=0}^q \beta_k z^k$. Thus finding $\boldsymbol{\beta}$ and σ^2 is in fact finding a factorization of Γ . This factorization is usually referred to as the Cramer-Wold factorization. There are two well known facts about this problem (see Wilson (1979)):

1. There is no real solution unless the spectral density $f(\omega) = \Gamma(e^{2\pi i \omega}) \geq 0$ for all $\omega \in [0, 1]$.
2. If $f(\omega) \geq 0$, then there are 2^q sets of MA coefficients that all give the same covariances, but only one of these sets will be such that $h(z)$ has all of its zeros outside the unit circle. This factor is called the Hurwitz factor of Γ .

Bauer's Algorithm

One solution to the problem of finding the Hurwitz factor is due to Bauer (1955). We have from Theorem 2.6.1 that if we let

$$\Gamma_M = \text{Toepl}(R(0), \dots, R(q), 0, \dots, 0) = \mathbf{L}_M \mathbf{D}_M \mathbf{L}_M^T,$$

then

$$\lim_{k \rightarrow \infty} L_{k,k-j} = \beta_j$$

$$\lim_{k \rightarrow \infty} D_{kk} = \sigma^2.$$

In Example 2.8, we give a macro illustrating this algorithm. Unfortunately, Bauer's algorithm can converge very slowly (see Pagano (1976)); that is, the elements of \mathbf{L}_M can take a long time to converge to the β 's. Further, there is no easily applied check for determining if the factorization is feasible, that is, if convergence can be reached at all.

Wilson's Algorithm

The algorithm derived by Wilson (1969,1979) converges rapidly and does have an easily applied check for the feasibility of the factorization. Wilson's algorithm is a straightforward application of the Newton-Raphson method for solving for the parameters σ^2 and β_1, \dots, β_q in

$$c_v = \sigma^2 \sum_{k=0}^{q-|v|} \beta_k \beta_{k+|v|} - R(v) = 0, \quad v = 0, \dots, q,$$

or, if we let $\tau_j = \sigma \beta_j$, $j = 0, \dots, q$, for solving for τ_0, \dots, τ_q in

$$c_v = \sum_{k=0}^{q-|v|} \tau_k \tau_{k+|v|} - R(v) = 0, \quad v = 0, \dots, q.$$

Wilson suggests using $\tau_{0,0} = \sqrt{R(0)}$ and $\tau_{0,j} = R(j)/\tau_{0,0}$, $j = 1, \dots, q$, as the initial approximation τ_0 to $\tau = (\tau_0, \dots, \tau_q)^T$. Then the Newton-Raphson iteration step is given by

$$\tau_{i+1} = \tau_i - \mathbf{G}_i^{-1} \mathbf{c}_i,$$

where

$$c_{i,v} = \sum_{k=0}^{q-|v|} \tau_{i,k} \tau_{i,k+|v|} - R(v)$$

is the v th element of \mathbf{c}_i and \mathbf{G}_i is the $(q+1 \times q+1)$ matrix (evaluated at τ_i) whose (j, k) th element is $\partial C_j / \partial \tau_k$. It is easy to see that $\mathbf{G}_i = \mathbf{T}_{1,i} + \mathbf{T}_{2,i}$ where

$$\mathbf{T}_{1,i} = \begin{bmatrix} \tau_{i,0} & \tau_{i,1} & \cdots & \cdots & \tau_{i,q} \\ \tau_{i,1} & \tau_{i,2} & \cdots & \tau_{i,q} & \\ \vdots & & \ddots & & \\ \vdots & & & \ddots & \\ \tau_{i,q} & & & & \end{bmatrix}$$

and

$$\mathbf{T}_{2,i} = \begin{bmatrix} \tau_{i,0} & \tau_{i,1} & \cdots & \cdots & \tau_{i,q} \\ & \tau_{i,0} & \cdots & \cdots & \tau_{i,q-1} \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & & & & \tau_{i,0} \end{bmatrix}.$$

Thus each iteration consists of finding $\tau_{i+1} = \tau_i - \mathbf{d}_i$ where $\mathbf{d}_i = \mathbf{G}_i^{-1} \mathbf{c}_i$ is found by solving the $(q+1 \times q+1)$ system of equations $\mathbf{G}_i \mathbf{d}_i = \mathbf{c}_i$. In general, solving such a set of equations requires on the order of q^3 operations (multiplications and additions). We next show how this system of equations can be solved in order q^2 operations.

Derivation of the Updating Step in Wilson's Algorithm

The key to Wilson's new (1979) updating step in the Newton-Raphson algorithm is to note that there is a Levinson-type recursion available to obtain τ_{i+1} from τ_i .

We first derive a polynomial equation that is equivalent to the system of equations that we want to solve. We have from $\tau_{i+1} = \tau_i - \mathbf{G}_i^{-1} \mathbf{c}_i$ and $\mathbf{G}_i = \mathbf{T}_{1,i} + \mathbf{T}_{2,i}$ that

$$\mathbf{T}_{1,i} \tau_{i+1} + \mathbf{T}_{2,i} \tau_{i+1} = \mathbf{T}_{1,i} \tau_i + \mathbf{T}_{2,i} \tau_i - \mathbf{c}_i.$$

But it is easy to see by the form of $\mathbf{T}_{2,i}$ that $\mathbf{c}_i = \mathbf{T}_{2,i} \tau_i - \mathbf{R}$, where $\mathbf{R} = (R(0), \dots, R(q))^T$, which gives

$$\mathbf{T}_{1,i} \tau_{i+1} + \mathbf{T}_{2,i} \tau_{i+1} = \mathbf{T}_{1,i} \tau_i + \mathbf{R} = \mathbf{R}_i + \mathbf{R},$$

where $\mathbf{R}_i = (R_i(0), \dots, R_i(q))^T$ and

$$R_i(v) = \sum_{k=0}^{q-|v|} \tau_{i,k} \tau_{i,k+|v|}, \quad v = 0, \dots, q.$$

Now we note that two $(q+1)$ -dimensional vectors \mathbf{a} and \mathbf{b} are equal if and only if

$$a(z) = \sum_{k=-q}^q a_{|k|} z^k = \sum_{k=-q}^q b_{|k|} z^k = b(z), \quad z \in \mathcal{C}.$$

If we let $\mathbf{a} = \mathbf{T}_{1,i} \boldsymbol{\tau}_{i+1}$, then $a(z) = c(z)d(z^{-1})$, where

$$c(z) = \sum_{k=0}^q \tau_{i,k} z^k \quad \text{and} \quad d(z) = \sum_{k=0}^q \tau_{i+1,k} z^k.$$

Thus we have

$$c(z)d(z^{-1}) + c(z^{-1})d(z) = e(z),$$

where

$$e(z) = \sum_{k=-q}^q (R_i(k) + R(k)) z^k.$$

We know the coefficients of $c(z)$ and $e(z)$ and would like to find those of $d(z)$.

If we let $f(z) = c(z)/\tau_{i,0}$ and $r(z) = e(z)/\tau_{i+1,0}$, we can write

$$f(z)d(z^{-1}) + f(z^{-1})d(z) = r(z),$$

where the coefficients f_1, \dots, f_q of $f(z)$ and those of $r(z)$ are known, the coefficient of z^0 in $f(z)$ is one, and we seek the coefficients of $d(z)$, that is, τ_{i+1} . This is the polynomial equation that we referred to above.

Define the coefficients $f_{k,j}$ for $1 \leq j \leq k \leq q$, by performing Levinson's algorithm backward starting from $f_{q,j} = f_j = \tau_{i,j}/\tau_{i,0}$, $j = 1, \dots, q$; that is,

$$f_{k,j} = \frac{f_{k+1,j} - f_{k+1,k+1} f_{k+1,k+1-j}}{1 - f_{k+1,k+1}^2}, \quad j = 1, \dots, q,$$

for $k = q-1, \dots, 1$. Recall that this means that

$$f_{k+1,j} = f_{k,j} + f_{k+1,k+1} f_{k,k+1-j}.$$

Thus if we define the polynomial $f_k(z) = \sum_{j=0}^k f_{k,j} z^j$, we have

$$f_{k+1}(z) = f_k(z) + f_{k+1,k+1} z^{k+1} f_k(z^{-1}).$$

Note that $f(z) = f_q(z)$ and thus we can write (letting $k+1 = q$)

$$\begin{aligned} r(z) &= \{f_{q-1}(z) + f_{q,q} z^q f_{q-1}(z^{-1})\} d(z^{-1}) \\ &\quad + \{f_{q-1}(z^{-1}) + f_{q,q} z^{-q} f_{q-1}(z)\} d(z) \\ &= f_{q-1}(z) \{d(z^{-1}) + f_{q,q} z^{-q} d(z)\} \\ &\quad + f_{q-1}(z^{-1}) \{d(z) + f_{q,q} z^q d(z^{-1})\} \\ &= f_{q-1}(z) h_q(z^{-1}) + f_{q-1}(z^{-1}) h_q(z), \end{aligned}$$

where $h_q(z) = d(z) + f_{q,q}z^q d(z^{-1})$ is a polynomial of degree q .

The highest power of z in $f_{q-1}(z)h_q(z^{-1})$ is $q-1$, while in $f_{q-1}(z^{-1})h_q(z)$ it is q which has coefficient equal to that of z^q in $h_q(z)$, which we will call γ_q , which has value $\tau_{i+1,q} + f_{q,q}\tau_{i+1,0}$. But γ_q is also equal to the coefficient of z^q in $r(z)$, that is, $\gamma_q = (R_i(q) + R(q)) / \tau_{i,0}$.

Now define the polynomial $d_{q-1}(z)$ of degree $q-1$ having coefficients $d_{q-1,1}, \dots, d_{q-1,q-1}$ and satisfying

$$d_{q-1}(z) = h_q(z) - \gamma_q z^q = d(z) + f_{q,q}z^q d(z^{-1}) - \gamma_q z^q.$$

Thus

$$d_{q-1,j} = d_{q,j} + f_{q,q}d_{q,q-j}, \quad j = 0, \dots, q-1,$$

where the coefficients of $d(z)$ (which are what we want) are denoted by $d_{q,j}$. Replacing j by $q-j$ in this equation gives

$$d_{q-1,q-j} = d_{q,q-j} + f_{q,q}d_{q,j}, \quad j = 0, \dots, q-1,$$

and if we solve this for $d_{q,j}$ and substitute the result into the equation for $d_{q-1,j}$ we get

$$d_{q,j} = \frac{d_{q-1,j} - f_{q,q}d_{q-1,q-j}}{1 - f_{q,q}^2}, \quad j = 0, \dots, q-1.$$

Except for $j = q$, this is a recursion for the d 's. We also have that γ_q is the coefficient of $h_q(z) = d(z) + f_{q,q}z^q d(z^{-1})$, that is, $\gamma_q = d_{q,q} + f_{q,q}d_{q,0}$, which means that if we define $d_{q-1,q} = \gamma_q$, then the above equation for $d_{q,j}$ holds for $j = q$ as well.

The next step in deriving the algorithm is to note that substituting $h_q(z) = d_{q-1}(z) + \gamma_q z^q$ into the equation for $r(z)$ gives

$$f_{q-1}(z)d_{q-1}(z^{-1}) + f_{q-1}(z^{-1})d_{q-1}(z) = r_{q-1}(z),$$

where $r_{q-1} = r(z) - \gamma_q z^{-q} f_{q-1}(z) - \gamma_q z^q f_{q-1}(z^{-1})$ has powers of z from $-(q-1)$ to $q-1$. This equation is identical in form to the one we started with except that the degrees have been reduced from q to $q-1$. The function $r_{q-1}(z)$ has coefficients

$$r_{q-1,j} = r_{q-1,-j}, \quad j = 0, \dots, q-1,$$

where $r_{q-1,0} = r_{q,0}$ and

$$r_{q-1,j} = r_{q,j} - \gamma_q f_{q-1,q-j}, \quad j = 1, \dots, q-1,$$

if we define $r_{q,0}, \dots, r_{q,q}$ to be the coefficients of $r(z)$.

We can now repeat this process until we get

$$f_0(z)d_0(z^{-1}) + f_0(z^{-1})d_0(z) = r_0(z),$$

that is, $d_{0,0} = r_{0,0}/2$.

Summary of the Updating Step

Thus we have the following algorithm for obtaining τ_{i+1} from τ_i . First find

$$f_{q,j} = \frac{\tau_{i,j}}{\tau_{i,0}}, \quad j = 1, \dots, q$$

$$r_{q,j} = \frac{R(j) + \sum_{k=0}^{q-|j|} \tau_{i,k} \tau_{i,k+|j|}}{\tau_{i,0}}, \quad j = 0, \dots, q.$$

Then for $k = q, \dots, 1$:

$$\gamma_k = r_{k,k}$$

$$f_{k-1,j} = \frac{f_{k,j} - f_{k,k} f_{k,k-j}}{1 - f_{k,k}^2}, \quad j = 1, \dots, k-1$$

$$r_{k-1,j} = r_{k,j} - \gamma_k f_{k-1,k-j}, \quad j = 1, \dots, k-1.$$

Finally, $d_{0,0} = r_{q,0}/2$, and for $k = 0, \dots, q-1$:

$$d_{k,k+1} = \gamma_{k+1}$$

$$d_{k+1,j} = \frac{d_{k,j} - f_{k+1,k+1} d_{k,k+1-j}}{1 - f_{k+1,k+1}^2}, \quad j = 0, \dots, k+1,$$

and $\tau_{i+1,j} = d_{q,j}$, $j = 0, \dots, q$.

One final point is that Wilson shows that the Cramer-Wold factorization is not feasible either if $\tau_{i,0}$ is negative for any iteration or if one of the $f_{k,k}$'s (which are analogous to partial autocorrelations) is not in $(-1, 1)$.

The CORRMA Command

The command

beta=CORRMA(**rho,q,R0,maxit,del,ier,rvar**)

performs the algorithm that we have just described. The input is the order **q**, the array **rho** which contains $\rho(1), \dots, \rho(q)$, the variance **R0**, the integer **maxit** which is the maximum number of Newton-Raphson steps that the user will allow, and the real scalar **del** which is a convergence criterion. Iterations continue until either

$$\max_{j=0, \dots, q} \frac{|\tau_{i+1,j} - \tau_{i,j}|}{|\tau_{i,j}|} < \mathbf{del},$$

in which case the output integer variable **ier** is set equal to 0 and **beta** and **rvar** are formed, or the maximum number of iterations is reached, in which

case **ier** is set equal to 1, and **beta** and **rvar** are not formed. Finally, if at any iteration, the factorization is deemed unfeasible, **ier** is set equal to 2, and **CORRMA** aborts. For most cases, a value of 20 or 30 for **maxit** with **del** = 10^{-5} should give satisfactory results.

2.6.4. Factoring an ARMA Covariance Generating Function

Suppose we are given the first $M+1$ autocovariances $R(0), R(1), \dots, R(M)$ of an $\text{ARMA}(p, q, \alpha, \beta, \sigma^2)$ process, and we would like to determine the coefficients and noise variance for the process. Note that this problem is essentially finding the factors $g(z)$ and $h(z)$ in the ARMA covariance generating function (see Theorem 2.5.6)

$$\Gamma(z) = \frac{h(z)h(z^{-1})}{g(z)g(z^{-1})}.$$

We know from Theorem 2.5.6 that we can obtain $\alpha_1, \dots, \alpha_p$ from the so-called high order Yule-Walker equations

$$\sum_{j=0}^p \alpha_j R(v-j) = 0, \quad v = q+1, \dots, q+p.$$

This results in the $(p \times p)$ system of equations

$$\begin{bmatrix} R(q) & \dots & R(q-p) \\ \vdots & & \vdots \\ R(q+p-1) & \dots & R(q) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_p \end{bmatrix} = - \begin{bmatrix} R(q+1) \\ \vdots \\ R(q+p) \end{bmatrix}.$$

Thus we need that $M \geq p+q$ in order to solve this system.

If we write

$$Y(t) = \sum_{j=0}^p \alpha_j X(t-j) = \sum_{k=0}^q \beta_k \epsilon(t-k),$$

then we have from $Y(t) = \sum_{k=0}^q \beta_k \epsilon(t-k)$ that Y is an $\text{MA}(q, \beta, \sigma^2)$ process, while from $Y(t) = \sum_{j=0}^p \alpha_j X(t-j)$ and the Filter Theorem, we have

$$\begin{aligned} R_Y(v) &= E(Y(t)Y(t+v)) \\ &= \sum_{j=0}^p \sum_{k=0}^p \alpha_j \alpha_k E(X(t-j)X(t-k)) \\ &= \sum_{j=0}^p \sum_{k=0}^p \alpha_j \alpha_k R(j+v-k), \quad v \geq 0. \end{aligned}$$

Thus we can calculate $R_Y(0), R_Y(1), \dots, R_Y(q)$ from this and then use Wilson's algorithm to find the β 's and σ^2 .

The CORRARMA Command

The command

```
rvar=CORRARMA(rho,R0,p,q,maxit,del,ier,alpha,beta)
```

performs the method described above. The input arguments **maxit** and **del** are the same as described in the **CORRMA** command, while the output integer **ier** is 0 if **CORRARMA** ends successfully, 1 if Wilson's algorithm does not converge, 2 if the factorization in Wilson's algorithm is judged infeasible, and 3 if the matrix in the high order Yule-Walker equations is judged to be singular.

The ARMACORR Command

Suppose $X \sim \text{ARMA}(p, q, \alpha, \beta, \sigma^2)$ and we would like to find the corresponding autocovariances $R(0), R(1), \dots, R(M)$. We describe the algorithm given by Wilson (1979).

If $M > q$, we can use the fact (see Theorem 2.5.6) that

$$R(v) = - \sum_{j=1}^p \alpha_j R(v-j), \quad v > q,$$

to find $R(q+1), \dots, R(M)$ if we first find $R(q-p), \dots, R(q)$. In fact we will find $R(0), \dots, R(r)$ for $r = \max(p, q)$ as this will guarantee that we will have all of the R 's that we need in order to use the difference equation (recall that $R(v) = R(-v)$).

The covariance generating function $\Gamma(z)$ of X is given by

$$\Gamma(z) = \sigma^2 \frac{h(z)h(z^{-1})}{g(z)g(z^{-1})},$$

where $g(z) = \sum_{j=0}^p \alpha_j z^j$ and $h(z) = \sum_{k=0}^q \beta_k z^k$. If we can find a polynomial γ of degree r having coefficients $\gamma_0, \dots, \gamma_r$ such that

$$\delta(z) = h(z)h(z^{-1}) = g(z)\gamma(z^{-1}) + g(z^{-1})\gamma(z),$$

then

$$\frac{1}{\sigma^2} \Gamma(z) = \frac{\gamma(z^{-1})}{g(z^{-1})} + \frac{\gamma(z)}{g(z)},$$

and we can determine $R(0), R(1), \dots, R(r)$ by equating coefficients of like powers of z on both sides of this equation. Since the zeros of g are all outside the

unit circle, then $1/g(z)$ and thus $\theta(z) = \gamma(z)/g(z)$ has only nonnegative powers of z . Similarly, $\gamma(z^{-1})/g(z^{-1})$ has only nonpositive powers of z . Thus we have

$$R(v) = \sigma^2 \theta_v, \quad v > 0,$$

where θ_v is the coefficient of z^v in $\gamma(z)/g(z)$, while $R(0) = 2\sigma^2 \theta_0$, since θ_0 is the coefficient of z^0 in both $\gamma(z^{-1})/g(z^{-1})$ and $\gamma(z)/g(z)$. Now $\theta(z) = \gamma(z)/g(z)$ means that $g(z)\theta(z) = \gamma(z)$; that is,

$$\sum_{j=0}^p \sum_{k=0}^{\infty} \alpha_j \theta_k z^{j+k} = \sum_{l=0}^r \gamma_l z^l.$$

The coefficient of z^s on the left hand side of this equation is $\sum_{j=0}^{\min(s,p)} \alpha_j \theta_{s-j}$, while on the right hand side it is γ_s . This means that $\theta_0 = \gamma_0$ and

$$\theta_s = \gamma_s - \sum_{j=1}^{\min(s,p)} \alpha_j \theta_{s-j}, \quad s = 1, \dots, r.$$

Thus it remains to find $\gamma_0, \dots, \gamma_r$ such that

$$\delta(z) = h(z)h(z^{-1}) = g(z)\gamma(z^{-1}) + g(z^{-1})\gamma(z).$$

But this equation is identical to the equation

$$r(z) = f(z)d(z^{-1}) + f(z^{-1})d(z)$$

in Section 2.6.3, except that the degrees of the polynomials f and d there were q , while in the present case, g is of degree p while γ is of degree r . Thus if $r = p$, that is, $p \geq q$, then the same algorithm as in Section 2.6.3 can be used with p replacing q and the coefficients of $\delta(z)$ and $g(z)$ replacing those of $r(z)$ and $f(z)$. If $p < q$, we can still use the algorithm by using $f_{k,j} = 0$ if $k > p$, and $d_{k+1,j} = d_{k,j}$ for $k \geq p$.

The command

```
rho=ARMACORR(alpha,beta,p,q,rvar,M,R0,ier)
```

produces the variance $R(0)$ and autocorrelations $\rho(1), \dots, \rho(M)$ in the scalar $R0$ and the array ρ . The output integer ier is 0 if **ARMACORR** finishes normally and is 1 if the process is judged to be nonstationary or noninvertible.

2.7. Examples and Problems

Example 2.1

EIGENVALUES OF TOEPLITZ MATRICES

There are situations in time series analysis where a knowledge of the eigenvalues of an $(n \times n)$ covariance matrix is useful (see part (g) of Theorem A.4.2). In this example we illustrate the fact that these eigenvalues are related to the values of the spectral density at the n frequencies $(j-1)/n$ for $j = 1, \dots, n$.

Let R and f be the autocovariance and spectral density functions of a covariance stationary time series and let

$$\lambda_{n,1} \geq \lambda_{n,2} \geq \dots \geq \lambda_{n,n}$$

be the eigenvalues of $\Gamma_n = \text{Toepl}(R(0), R(1), \dots, R(n-1))$ arranged in nonincreasing order. Suppose that m is the largest number such that $f(\omega) \geq m$ for all ω and M is the smallest number such that $f(\omega) \leq M$ for all ω . Suppose that $M < \infty$ and let h be any continuous function defined on the interval $[m, M]$. Let

$$f_{(1)} \geq f_{(2)} \geq \dots \geq f_{(n)}$$

be the values of f (arranged in nonincreasing order) at the frequencies $(j-1)/n$ for $j = 1, \dots, n$. Then Grenander and Szegö (1958, p. 65) show that

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{v=1}^n [h(\lambda_{n,v}) - h(f_{(v)})] = 0$$

and

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{v=1}^n h(\lambda_{n,v}) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{v=1}^n h(f_{(v)}) = \int_0^1 h(f(\omega)) d\omega.$$

Thus if h is the identity function, we have that the eigenvalues and the sorted values of the spectral density are essentially the same for large n . Further, the integral of any continuous function of the spectral density of f (for example $h(x) = x^2$ or $h(x) = \log x$) is also related to the eigenvalues of Γ_n .

The following macro will create and list the eigenvalues and sorted values of the spectral density for a user-specified AR(p) process.

```

1 ;;
2 ;;   EIG.MAC: macro to illustrate the fact that the eigenvalues
3 ;;           of a covariance matrix are related to the spectral
4 ;;           density of the related process.
5 ;;
6 ;;   INPUT: p, alpha (order and coefficients of the AR process to
7 ;;           use for illustration)

```

```

8 ;;          n (size of covariance matrix to use. n must be .le.60)
9 ;;
10 PAUSE
11 ;start
12 IF(n.le.60,s1)
13 ;;
14 ;;  n must be less than or equal to 60
15 ;;
16 GOTO(end)
17 ;
18 ;s1
19 ;
20 rho=ARCORR(alpha,p,1,n,r0,ier)
21 r=rho*r0
22 gam=TOEPL(r,r0,n)
23 ;;
24 ;;    Finding the eigenvalues takes some time....
25 ;;
26 evals=EIG(gam,n,100,ier)
27 f=ARSP(alpha,p,1,n)
28 f1=f          ;f1 is just here so we can use DOUBLE
29 DOUBLE(f,f1,n) ;form f at all n frequencies
30 f=SORT(f,n)
31 f=REVERSE(f,n) ;now f has spectra in descending order
32 PROMPTOFF
33 LIST(evals)
34 LABEL(f)='Values of f in descending order'
35 LIST(f)
36 PROMPTON
37 ;
38 ;end
39 ;

```

To illustrate the use of this macro, Table 2.2 gives the result of using it for the AR(2) process having coefficients 0.3 and 0.9 (see Figure 2.2 for a plot of the log of the spectral density of this process). We have found the eigenvalues of the (40×40) covariance matrix of the process. Note the close agreement between the eigenvalues and the spectral density except for the eight largest values.

Example 2.2

THE SPECTRAL DENSITY FUNCTION

We saw in Section 2.2 that the spectral density function of a time series can be thought of as the average over many long realizations of the periodogram. The PERAVE.MAC macro can be used to generate a series of realizations from a specified autoregressive process and keep a running average of the periodograms found for each one. There are four plots on the screen while the macro is running: (1) the true spectral density of the process, (2) the current data set, (3) the periodogram of the current data set, and (4) the average of the periodograms that have been produced. The point of the macro is that the

Table 2.2. An Example of the Relationship Between the Eigenvalues of a Toeplitz Matrix and the Corresponding Spectral Density

Eigenvalues					
1	60.749760	60.589540	21.689460	21.507380	8.979041
6	8.837506	4.657563	4.537861	2.829798	2.721278
11	1.912571	1.811326	1.392952	1.298594	1.071119
16	.986147	.856760	.787423	.703464	.658951
21	.586699	.573033	.512443	.497300	.467412
26	.442225	.421246	.417380	.401766	.393535
31	.369590	.328809	.296827	.271782	.252204
36	.237020	.225453	.216951	.211127	.207729
Values of f in descending order					
1	102.427900	102.427900	10.930160	10.930160	10.000000
6	10.000000	3.082259	3.082259	2.729008	2.729008
11	1.484353	1.484353	1.256205	1.256205	.914162
16	.914162	.735545	.735545	.651911	.651911
21	.514863	.514863	.496563	.496563	.440136
26	.440136	.402263	.402263	.390625	.369535
31	.369535	.295989	.295989	.251651	.251651
36	.225203	.225203	.211065	.211065	.206612

average of the periodograms gets closer and closer to the true spectral density as the number of realizations increases. Note that all of the spectral plots are on a log scale.

```

1 ;;
2 ;;   PERAVE: macro to illustrate the unbiasedness of the
3 ;;       periodogram. For a specified AR(p) process,
4 ;;       the macro generates nsamps samples of size n.
5 ;;
6 ;;       On the screen at the same time, the true AR
7 ;;       spectral density is shown together with the
8 ;;       sample, the periodogram of the sample, and the
9 ;;       average of all the periodograms so far.
10 ;;
11 ;;   INPUT: n (sample size), p, alpha, nsamps
12 ;;           p, alpha (order and coefficients of AR process)
13 ;;           nsamps (number of samples)
14 ;;           seed (random number generator seed)
15 ;;
16 PAUSE
17 ;start
18 PLOTOM           ;switch to graphics mode
19 BATCHOM         ;no pause between graphs
20 ns=1            ;initialize sample counter
21 x=ARDT(alpha,p,1,seed,n,ier,r0true) ;find variance of process
22 f=ARSP(alpha,p,1,n) ;find true f
23 LABEL(f)='True AR(#p#) Spectral Density'
24 PLOTSIZE(1)      ;true spectra goes in upper left corner
25 PLOTSP(f,n,r0true) ;plot it

```

```

26 q={n/2}+1
27 fave=LINE(q,0,0) ;initialize average periodogram
28 ;
29 ;startloop
30 ;
31 ERASE(2) ;erase old data
32 PLOTSIZE(2) ;data go in lower left
33 x=ARDT(alpha,p,1,0,n,ier,r0)
34 LABEL(x)='Sample Number #ns#'
35 PLOT(x,n) ;plot it
36 ERASE(3) ;erase old periodogram
37 PLOTSIZE(3) ;periodogram goes in upper right
38 rho=CORR(x,n,0,n,1,r0,fh)
39 LABEL(fh)='Periodogram for Sample #ns#'
40 PLOTSP(fh,n,r0) ;plot it
41 fave={({ns-1}*fave/ns)+{fh/ns} ;find average
42 ERASE(4) ;erase old average
43 PLOTSIZE(4) ;average goes in lower right
44 LABEL(fave)='Average of #ns# Samples'
45 PLOTSP(fave,n,r0true)
46 if(ns.eq.nsamps,endoloop) ;all finished?
47 ns=ns+1 ;no
48 GOTO(startloop)
49 ;
50 ;endoloop
51 ;
52 BATCHOFF ;yes
53 GRMENU(0,.5,-6,6) ;let user FIND, PRINT, etc

```

Example 2.3

THE DIRICHLET KERNEL

The macro DIRICH.MAC given below was used to generate Figure 2.3.

```

1 ;;
2 ;;   DIRICH.MAC: macro to calculate the Dirichlet kernel for
3 ;;           user specified M for Q frequencies between
4 ;;           frequencies f1 and f2 (inclusive). The
5 ;;           kernel is then plotted with the vertical
6 ;;           scale being between ymin and ymax.
7 ;;
8 ;;   INPUT: M, Q (number of frequencies between f1 and f2)
9 ;;           f1, f2, ymin, ymax (these can be entered as zeros)
10 ;;
11 ;;
12 PAUSE
13 ;start
14 freq=LINE(Q,f1,f2,1) ;freq contains frequencies
15 ff=pi*freq
16 ff=REPLACE(ff,Q,4,0,0,.0001) ;make sure frequency zero not included
17 den=SIN(ff,Q) ;denominator
18 om=2.*M+1

```

```

19 ff=om*ff
20 num=SIN(ff,Q)           ;numerator
21 dir=num/den
22 LABEL(dir)='Dirichlet Kernel For M=###'
23 LABEL(freq)=' '
24 PLOT(freq,dir,Q,f1,f2,ymin,ymax)

```

Example 2.4 THE BANDPASS FILTER EXAMPLE

The macros BAND.MAC and PARZ.MAC were used to form the graphs in Figure 2.4.

```

1 ;;
2 ;;  BAND.MAC: macro to approximate the ideal transfer function
3 ;;  of a bandpass filter centered at frequency .25
4 ;;  and bandwidth .1.
5 ;;
6 ;;  INPUT: m (number of terms in the approximation)
7 ;;
8 PAUSE
9 ;start
10 ;
11 ;  calculate coefficients:
12 ;
13 u=.25
14 v=.1
15 b0=4.*v
16 ok=LINE(m,0,1)           ; ok is 1,...,m
17 argc=2*pi*ok*u
18 args=2*pi*ok*v
19 c=COS(argc,m)
20 s=SIN(args,m)
21 b=2*c*s/(pi*ok)
22 b=<b0,b>                 ; b contains the m+1 coefficients
23 MACRO(parz,start)        ; The PARZ macro gets the m+1 Parzen weights
24 bp=b*z                   ; bp has weighted coefficients
25 nmm=200-m-1              ; We'll evaluate the transfer functions
26 zero=LINE(nmm,0,0)       ; at 200 points in 0 to 1 and plot
27 b=<b,zero>                ; at the 101 points in 0 to .5.
28 bp=<bp,zero>
29 ;
30 ;  Do unweighted transfer function and plot:
31 ;
32 FFT(b,200,101,1,zr,zi)   ; Sum for indices 0 through m
33 FFT(b,200,101,-1,zr1,zi1) ; Sum for indices -m through 0
34 zr=zr+zr1-b0             ; Add together and subtract 0 index term
35 x=LINE(101,-.005,.005)   ; Now get ready for plot:
36 x1=<-.15,.15,.35,.35>
37 h1=<0,1,1,0>
38 x2=<0,.5>
39 h2=<0.0,0.0>

```

```

40 xx=<x1,x2,x>
41 yy=<h1,h2,zr>
42 type=<2,2,2>
43 n=<4,2,101>
44 LABEL(xx)= ' '
45 LABEL(yy)= 'No weights, M=#m#'
46 PLOTK(xx,yy,n,3,type,0,.5,-.2,1.20) ;plot the unweighted approximation
47 ;
48 ; Weighted Version:
49 ;
50 FFT(bp,200,101,1,zr,zi)
51 FFT(bp,200,101,-1,zr1,zi1)
52 zr=zr+zr1-b0
53 yy=<h1,h2,zr>
54 LABEL(xx)= ' '
55 LABEL(yy)= 'Parzen weights, M=#m#'
56 PLOTK(xx,yy,n,3,type,0,.5,-.20,1.20) ;plot the weighted approximation

1 ;;
2 ;; PARZ.MAC: macro to calculate the Parzen weight function for
3 ;; index 0,1,...,m and return it in z[1],...,z[m+1].
4 ;;
5 ;; INPUT: m (must be even)
6 ;;
7 PAUSE
8 ;start
9 z=LINE(m,0,1)
10 z=z/m
11 mb2=m/2
12 mb2p1=mb2+1
13 z1=EXTRACT(z,1,mb2) ;z1 will get first half, z2 gets second half
14 z2=EXTRACT(z,mb2p1,m)
15 z1=1-6*z1*(z-z^2)
16 z2=2*(1-z2)^3
17 z=<1,z1,z2> ; z(0)=1
18 CLEAN(z1,z2,mb2,mb2p1) ;get rid of stuff we don't need

```

Example 2.5	RANDOM WALKS
--------------------	---------------------

The RW.MAC was used (with n=200) to produce the random walk series in Figure 2.5.

```

1 ;;
2 ;; RW.MAC: macro to plot 5 random walks of length n on the
3 ;; screen at once.
4 ;;
5 ;; INPUT: n (length of each random walk)
6 ;; seed (random number generator seed)
7 ;; dist (which distribution to use in WW)

```

```

8 ;;
9 PAUSE
10 ;start
11 PROMPTOFF
12 y=wn(seed,10) ;warm up random number generator
13 sample#=1
14 ;
15 ;s1
16 ;
17 LIST(sample#)
18 y1=WN(0,n,dist)
19 y1=CUM(y1,n,1) ;get next random walk
20 y=<y1,y> ;append this one to the others
21 IF(sample#.eq.5,plot) ;have all 5?
22 sample#=sample#+1 ;no
23 GOTO(s1)
24 ;
25 ;plot ;yes
26 ;
27 x=LINE(n,0,1)
28 x=<x,x,x,x,x>
29 type=<2,2,2,2,2>
30 LABEL(y)='5 Random Walks (n=#n#, dist=#dist#)'
31 LABEL(x)=' '
32 PLOTK(x,y,n,5,type)
33 PROMPTON

```

Example 2.6 GENERATING STABLE COEFFICIENTS

We are often interested in obtaining sets of coefficients whose characteristic polynomial has all of its roots outside the unit circle. The macro `RANDCOEF.MAC` given in this example will randomly generate such a set of coefficients. First the `WN` command is used to find a set of numbers between 0 and 1. These are then converted to numbers between -1 and 1 , which we then treat as partial autocorrelations. From these partials we can find the corresponding autoregressive coefficients. These coefficients can be thought of as either AR, MA, or one part of a set of ARMA coefficients.

```

1 ;;
2 ;; RANDCOEF.MAC: macro to generate a set of partial
3 ;; autocorrelations that are uniformly
4 ;; distributed in the interval (-1,1)
5 ;; and then find the corresponding AR
6 ;; coefficients. This can be used to
7 ;; generate stable sets of AR, MA, or ARMA
8 ;; coefficients.
9 ;;
10 ;; INPUT: seed (seed for random number generator)
11 ;; p (number of coefficients to generate)
12 ;;

```

```

13 ;;   OUTPUT: part (partials), coeffs (coefficients)
14 ;;
15 PAUSE
16 ;start
17 part=WH(seed,p,2)           ;U(0,1)
18 part=2*{part-.5}           ;U(-1,1)
19 coeffs=PARTAR(part,p) ;find coefficients
20 LABEL(part)='Partial Autocorrelations'
21 LABEL(coeffs)='Coefficients'

```

Example 2.7 PATTERNS IN ARMA PROCESSES

We have seen that the correlogram, partial correlogram, and spectral density functions of AR, MA, and ARMA processes have certain distinguishing characteristics. In this example we give a macro that can be used to study these characteristics. The macro is called IDARMA and can be used for AR, MA, or ARMA processes by specifying appropriate values for the orders of an ARMA process ($p=0$ for an MA process and $q=0$ for an AR process). In each case, a series of coefficients are randomly generated, and for each set the correlogram, partial correlogram, spectral density, and one realization of length 200 are displayed. The macro has an entry point labeled ;known that can be used if the user wants to generate the above quantities for a specified model and coefficients.

```

1 ;;
2 ;;   IDARMA.MAC: Macro to illustrate the patterns in various diagnostics
3 ;;               for an ARMA(p,q) process. Successive coefficients are
4 ;;               randomly generated and the properties of the resulting
5 ;;               process displayed.
6 ;;
7 ;;   NOTE: This macro will execute until it is broken by the user.
8 ;;
9 ;;   INPUT: p (0 for MA processes), q (0 for AR processes)
10 ;;        seed (seed for random number generator)
11 ;;
12 PAUSE
13 ;start
14 CLS
15 part=WH(seed,10)           ;warm up generator
16 ;
17 ;   Randomly generate coefficients:
18 ;
19 ;startloop
20 ;
21 IF(p.gt.0)                ;skip this part if doing an MA process
22     part=WH(0,p,2)
23     part=2*{part-.5}       ;now elements of part are between -1 and 1
24     alpha=PARTAR(part,p)  ;now alpha are stable AR coefficients
25 ENDDIF

```



```

26 IF(q.gt.0)                                ;skip this part if doing an AR process
27     part=WM(0,q,2)
28     part=2*(part-.5)
29     beta=PARTAR(part,q)    ;now beta are invertible MA coefficients
30 ENDIF
31 ;
32 ;known
33 ;
34 IF(p.gt.0)
35     LABEL(alpha)='Autoregressive Coefficients'
36     POLYROOTS(alpha,1,p,100,rr,ri,ier)
37     rmod={rr^2+ri^2}^-.5
38     LIST(alpha)
39     LABEL(rmod)='Modulus of Roots'
40     LIST(rr)
41     LIST(ri)
42     LIST(rmod)
43 ENDIF
44 IF(q.gt.0)
45     LABEL(beta)='Moving Average Coefficients'
46     POLYROOTS(beta,1,q,100,rr,ri,ier)
47     rmod={rr^2+ri^2}^-.5
48     LIST(beta)
49     LABEL(rmod)='Modulus of Roots'
50     LIST(rr)
51     LIST(ri)
52     LIST(rmod)
53 ENDIF
54 ;
55 ;    Now do correlations, spectra, and data:
56 ;
57 pq=p*q
58 IF(pq.gt.0)
59     rho=ARMACORR(alpha,beta,p,q,1,30,r0,ier)
60     f=ARMASP(alpha,beta,p,q,1,256)
61     x=ARMADT(alpha,beta,p,q,1,0,200,30,rho,r0,ier)
62     GOTO(display)
63 ENDIF
64 IF(q.gt.0)
65     rho=MACORR(beta,q,1,30,r0)
66     f=MASP(beta,q,1,256)
67     x=MADT(beta,q,1,0,200)
68     GOTO(display)
69 ENDIF
70 IF(p.gt.0)
71     rho=ARCORR(alpha,p,1,30,r0,ier)
72     f=ARSP(alpha,p,1,256)
73     x=ARDT(alpha,p,1,0,200,ier,r0)
74 ENDIF
75 ;
76 ;display
77 ;
78 PAUSE
79 zero=LINE(30,0,0)

```

```

80 LABEL(rho)='Correlogram, R(0)=@r0@'
81 LABEL(zero)='      '
82 PLOT2(rho,zero,30,30,3,3,0,30,-1,1)
83 part=CORRAR(rho,r0,30,rvar)
84 part=ARPART(part,30,ier)
85 LABEL(part)='Partial Correlogram'
86 PLOT2(part,zero,30,30,3,3,0,30,-1,1)
87 PLOTSP(f,256,r0)
88 PLOT(x,200)
89 GOTO(startloop)

```

Example 2.8

BAUER'S ALGORITHM

The BAUER.MAC macro can be used to illustrate the structure of the modified Cholesky decomposition of the correlation matrix of an MA(q) process. Using the entry point labeled ;known allows the user to use the macro for specified set of coefficients.

```

1 ;;
2 ;;  BAUER.MAC: macro to illustrate Bauer's algorithm for
3 ;;                finding MA parameters from correlations.
4 ;;                The macro randomly generates MA coefficients,
5 ;;                forms the correlations, and then finds and lists the
6 ;;                modified Cholesky decomposition of the
7 ;;                correlation matrix of size M.
8 ;;
9 ;;  INPUT: q, M (order of MA process and size of correlation matrix)
10 ;;          seed (random number generator seed for generating beta)
11 ;;
12 PAUSE
13 ;start
14 beta=WM(seed,q,2)      ;U(0,1)
15 beta=2*(beta-.5)      ;U(-1,1)
16 beta=PARTAR(beta,q)   ;beta's are MA coefficients
17 ;
18 ;known                  ;use this entry point for known beta's
19 ;
20 LABEL(beta)='MA Coefficients'
21 rho=MACORR(beta,q,1,M,r0) ;get correlations
22 gam=TOEPL(rho,1,M)      ;form matrix
23 MCHOL(gam,M,L,D,ier)    ;decompose
24 L=TRANS(L,10,10)       ;we have to transpose L so that LIST works right
25 LABEL(L)='Lower Triangular Cholesky Factor'
26 M2=M*M
27 LIST(L,M2,M,10f6.3)
28 LIST(beta,q)

```

To illustrate the macro, we used the ;known entry point with an MA(3) process having coefficients 0.8, -0.75, and -0.8, and obtained the output given

Table 2.3. An Example of Bauer's Algorithm

Lower Triangular Cholesky Factor										
1		1.000	.000	.000	.000	.000	.000	.000	.000	.000
11		.281	1.000	.000	.000	.000	.000	.000	.000	.000
21		-.489	.455	1.000	.000	.000	.000	.000	.000	.000
31		-.281	-.445	.579	1.000	.000	.000	.000	.000	.000
41		.000	-.306	-.633	.668	1.000	.000	.000	.000	.000
51		.000	.000	-.494	-.596	.727	1.000	.000	.000	.000
61		.000	.000	.000	-.514	-.682	.767	1.000	.000	.000
71		.000	.000	.000	.000	-.637	-.656	.779	1.000	.000
81		.000	.000	.000	.000	.000	-.649	-.692	.786	1.000
91		.000	.000	.000	.000	.000	.000	-.713	-.684	.775
MA Coefficients										
1		.800000		-.750000		-.800000				

in Table 2.3. Note how the diagonal elements of L are converging (slowly) to the corresponding elements of the **beta** array.

Computational Problems

C2.1. Use the **MASP** command with $Q=256$ to find the square modulus of the frequency transfer function of the 12th difference filter. The input to **MASP** will be that of an **MA(12)** with all coefficients zero except $\beta_{12} = -1$. Use **PLOTSP** to plot the logarithm of the square modulus. Does the result correspond to what we found in Section 2.3 on the effect of differencing?

C2.2. Create a new macro called **RW1** by modifying the **RW** macro to use $U(-1,1)$ variables for the white noise process. Note that a realization from a time series process is also called a sample path. Do the sample paths for random walks using uniform white noise look much different than those using Gaussian white noise? Generate a plot of five sample paths of length 200 from each macro and compare them.

C2.3. What four sets of **MA(2)** parameters (coefficients and error variance) can lead to the autocovariances $R(0) = 1.3125$, $R(1) = 0.625$, and $R(2) = 0.25$? Which of these sets leads to a characteristic polynomial having all of its zeros outside the unit circle? Which of these sets does **TIMESLAB** pick if you use the **CORRMA** command? (Hint: See the section on nonidentifiability of **MA** processes.)

C2.4. If X is an AR(1) process with $\alpha = 0.5$, write $X(t)$ as a function of $X(t-2)$ and some ϵ 's and of $X(t+2)$ and some ϵ 's.

C2.5. Which of the following sets of coefficients can be regarded as the coefficients of an AR process?

- a) $\alpha = (-0.90, 0.80)$
- b) $\alpha = (-1.54, 0.40)$
- c) $\alpha = (1.0, -0.904, -0.70)$
- d) $\alpha = (-1.22, 1.16, -0.70)$
- e) $\alpha = (-0.09, -1.5884, -0.024, 0.90)$

C2.6. The macro ARGAMI.MAC will find the inverse of the $(n \times n)$ covariance matrix of an AR(p) process. Use it for $n = 10$ and for the AR(1) process having coefficient -0.5 and the AR(2) process having coefficients 0.3 and 0.9 . From these results, infer that Γ_n^{-1} is a banded matrix, that is, its (j, k) th element is zero for $|j - k| > p$. What else do you notice about the elements of the matrix?

```

1 ;;
2 ;;   ARGAMI.MAC: macro to find the inverse of the (n x n) Toeplitz
3 ;;               correlation matrix of an AR(p) process having
4 ;;               coefficients alpha and error variance 1.
5 ;;
6 ;;   INPUT: p, alpha, n (must be less than or equal to 10)
7 ;;
8 PAUSE
9 ;start
10 IF(n.le.10,s1)
11 ;;
12 ;;   n must be less than or equal to 10.
13 ;;
14 GOTO(end)
15 ;s1
16 rho=ARCORR(alpha,p,1,n,r0,ier)
17 gam=TOEPL(rho,1,n)
18 gami=MINV(gam,n,ier)
19 n2=n*n
20 LABEL(gami)='Inverse of Corr. Matrix, R(0)=@r0@'
21 LIST(gami,n2,n,10f6.2)
22 ;end

```

C2.7. Write a macro that will form the $(n \times n)$ permutation matrix.

C2.8. Write a macro that will generate a realization of length n from an ARMA process whose errors have any of the eight distributions in the **WN** command. (Hint: Use zeros as starting values and use **WN** and **FILT** to get the MA part and then use the difference equation form of **ARDT** to generate a realization of length $2n$. Use **EXTRACT** to extract the second half of this realization to use as the output realization of length n .)

C2.9. Using the **IDARMA** macro in Example 2.6, find an AR, MA, and ARMA process of each of the following kinds:

- a) An excess of high frequency.
- b) An excess of low frequency.
- c) A single peak in the spectral density.

Wait until you see a process having the desired characteristic and then reinvoke the macro with the **;known** entry point and get a hard copy version of the plots for that process.

C2.10. In this book, we consider primarily processes that can be expressed as a general linear process (see Section 2.5.9). Write a macro that will generate a realization of length n from the simple nonlinear process

$$X(t) = \epsilon(t) = \beta\epsilon(t-1)\epsilon(t-2),$$

where ϵ is a white noise process. See pages 866-890 of Priestley (1981) for a general discussion of nonlinear time series.

Theoretical Problems

T2.1. A time series X is said to be strictly stationary if for any positive integer n and any $n+1$ elements t_1, \dots, t_n and h of the index set T , the joint distributions of

$$X(t_1), \dots, X(t_n) \quad \text{and} \quad X(t_1+h), \dots, X(t_n+h)$$

are the same. Show (a) that a strictly stationary series is covariance stationary and (b) that a covariance stationary Gaussian process is strictly stationary. (Hint: See Section A.4.)

T2.2. Show that the sum and difference of two uncorrelated covariance stationary time series are also covariance stationary.

T2.3. In Figure 2.5, how did we know that the five random walk realizations would probably fit on a vertical axis ranging from -30 to 30 ?

T2.4. Show that the largest value that $\rho(1)$ can have for an MA(1) process is .5. Show that for an MA(q) process, the largest value that $\rho(1)$ can be is $\cos[\pi/(q+2)]$. Thus is there an MA(2) process having $\rho(1) = 0.8$?

T2.5. Write the spectral density f of an MA(3) process as a third-degree polynomial in $z = \cos \omega$, that is, as

$$f(\omega) = \theta_0 + \theta_1 z + \theta_2 z^2 + \theta_3 z^3$$

for some coefficients θ .

T2.6. Show that if $X \sim \text{AR}(p)$ and $Y \sim \text{WN}$, and X and Y are uncorrelated, then the spectral density function of the series $Z = X + Y$ is in the form of that of an ARMA(p, p) time series. See Granger and Morris (1976) and Engel (1984) for a general discussion of the properties of sums and products of ARMA processes.

T2.7. Use the results of Problem T1.12 to show that an AR(2) process will appear cyclic of period p if $\alpha_1 = -2\cos(2\pi/p)$ and $\alpha_2 \doteq 1$. (Hint: Show that an AR(2) process with these coefficients has a “signal to noise” ratio $R(0)/\sigma^2$ that is very large, and thus the AR difference equation is almost a deterministic homogeneous difference equation in the sense that the variability in ϵ is insignificant relative to that in X . See the **ARCORR** command in Section 2.6 for a discussion of the relationship between $R(0)$ and σ^2 .)

T2.8. We have defined the spectral density function on the interval $[0, 1]$. Many authors define it on the interval $[-\pi, \pi]$ as in the following. Let R be the autocovariance function of a covariance stationary time series X and suppose the spectral density of X exists. Show that we can write

$$R(v) = \int_{-\pi}^{\pi} f(\lambda) e^{iv\lambda}, \quad v \in \mathcal{Z},$$

where

$$f(\lambda) = \frac{1}{2\pi} \sum_{v=-\infty}^{\infty} R(v) e^{-iv\lambda}, \quad \lambda \in [-\pi, \pi].$$

T2.9. Prove part (c) of Theorem 2.3.1.

T2.10. Show that the following are true for integers j and k :

$$\min(j, k) = \frac{(j + k) - |j - k|}{2}$$

$$\max(j, k) = \frac{(j + k) + |j - k|}{2}.$$

T2.11. Show that the Dirichlet kernel

$$D_M(\omega) = \sum_{j=-M}^M e^{2\pi i j \omega}$$

discussed in Section 2.3 can be written as

$$D_M(\omega) = \frac{\sin[(M + \frac{1}{2})2\pi\omega]}{\sin \pi\omega}.$$

(Hint: Use a geometric series.)

T2.12. Verify that the expressions for the coefficients of the bandpass filter example in Section 2.3 are correct.

T2.13. Show that if \mathbf{P}_n is the $(n \times n)$ permutation matrix, and $\mathbf{\Gamma}_n$ is an $(n \times n)$ symmetric Toeplitz matrix, then

$$\mathbf{\Gamma}_n = \mathbf{P}_n \mathbf{\Gamma}_n \mathbf{P}_n \quad \text{and} \quad \mathbf{\Gamma}_n^{-1} = \mathbf{P}_n \mathbf{\Gamma}_n^{-1} \mathbf{P}_n.$$

(Hint: Use the fact that $\mathbf{P}_n^2 = \mathbf{I}_n$.)

T2.14. If ϵ is a white noise process with variance σ^2 , show that the process X defined by

$$X(t) = \sum_{j=0}^t \epsilon(j), \quad t \geq 0,$$

is a random walk process.

T2.15. Suppose that $X(0)$ is a random variable having mean μ_X and variance σ_X^2 . For $t > 0$ let

$$X(t) = -\alpha X(t-1) + \epsilon(t),$$

where $\epsilon \sim \text{WN}(\sigma^2)$ and $\epsilon(t)$ is uncorrelated with $X(0)$ for all t .

a) Show that

$$X(t) = (-\alpha)^t X(0) + \sum_{j=0}^{t-1} (-\alpha)^j \epsilon(t-j).$$

b) Thus show that

$$E(X(t)) = (-\alpha)^t \mu_X$$

$$\text{Cov}(X(t), X(t+v)) = (-\alpha)^{2t+v} \sigma_X^2 + \sigma^2 \sum_{j=0}^{t-1} (-\alpha)^{2j+v}.$$

c) If $|\alpha| < 1$, show that

$$\lim_{t \rightarrow \infty} E(X(t)) = 0$$

$$\lim_{t \rightarrow \infty} \text{Cov}(X(t), X(t+v)) = \frac{\sigma^2(-\alpha)^v}{1-\alpha^2}.$$

A time series (such as this one) whose means and covariances converge as $t \rightarrow \infty$ is said to be asymptotically covariance stationary.

d) Show that if we let μ_X and σ_X^2 be these limiting values of the mean and variance of $X(t)$, then X is covariance stationary with mean zero and autocovariance function

$$R(v) = \frac{\sigma^2(-\alpha)^v}{1-\alpha^2}.$$

Note that this device can be used to construct any autoregressive process as long as none of the zeros of the characteristic polynomial are on the unit circle. For an $\text{AR}(p)$ process, we would assign arbitrary means and covariances to p initial random variables, determine the long-run mean (which will be zero) and covariance function, and then assign these “long-run” moments to the initial variables.

T2.16. Show that a Gaussian $\text{AR}(1)$ process is a Markov process; that is, for any positive integer n and any $n+1$ integers

$$t_1 < \cdots < t_n < t_{n+1},$$

the conditional distributions of $X(t_{n+1})$ given the X 's at all of the first n times are the same as those of $X(t_{n+1})$ given just $X(t_n)$. (Hint: See part (c) of Theorem A.4.2 and part (a) of Theorem 2.6.5.)

T2.17. Use long division to find the coefficients of the $MA(\infty)$ representation of an $ARMA(1,1)$ process having AR coefficient α and MA coefficient β ; that is, divide $1 + \beta z$ by $1 + \alpha z$. Do the division in such a way that only nonnegative powers of z are in the answer. What happens for large powers of z if $|\alpha| > 1$?

T2.18. Let X and Y have the bivariate normal distribution with means both equal to zero, variances both equal to σ^2 , and correlation coefficient equal to ρ .

a) Show that

$$\Pr(X > 0, Y > 0) = \frac{1}{4} + \frac{1}{2\pi} \arcsin \rho.$$

(Hint: Use polar coordinates to do the integral of the joint pdf over the first quadrant.)

b) Now suppose that X is a Gaussian stationary time series with mean zero, and define the clipped time series Y by

$$Y(t) = \begin{cases} 1, & \text{if } X(t) > 0 \\ 0, & \text{if } X(t) \leq 0. \end{cases}$$

Use the result of part (a) to show that the autocorrelation function ρ_Y of Y is related to the autocorrelation function ρ_X of X by

$$\rho_Y(v) = \frac{2}{\pi} \arcsin \rho_X(v).$$

CHAPTER 3

Statistical Inference for Univariate Time Series

In this chapter we describe some of the basic statistical inferences that can be made about univariate time series. In Section 3.1 we give the sampling properties of the descriptive statistics that were introduced in Chapter 1. In Section 3.2 we describe two standard tests for white noise. In Section 3.3 we discuss the problem of estimating the spectral density of a time series without making any assumptions about the model generating the data. This is called nonparametric spectral density estimation. In Sections 3.4 and 3.5 we consider respectively the questions of estimating the parameters of an ARMA model and identifying the orders of the model. Then in Section 3.6 we describe the so-called Box-Jenkins forecasting procedure and in Section 3.7 some other modeling strategies. In Section 3.8 we discuss the use of ARMA models for spectral density estimation, and finally in Section 3.9 we consider searching for periodicities in data.

3.1. Sampling Properties of Descriptive Statistics

In Chapter 1 we introduced the statistics

$$\bar{X} = \frac{1}{n} \sum_{t=1}^n X(t)$$

$$\hat{R}(v) = \frac{1}{n} \sum_{t=1}^{n-|v|} (X(t) - \bar{X})(X(t+|v|) - \bar{X}), \quad |v| < n$$

$$\hat{\rho}(v) = \frac{\hat{R}(v)}{\hat{R}(0)}, \quad |v| < n$$

$$\hat{f}(\omega) = \frac{1}{n} \left| \sum_{t=1}^n (X(t) - \bar{X}) e^{2\pi i(t-1)\omega} \right|^2, \quad \omega \in [0, 1],$$

as well as the sample partial autocorrelation function. We will now use upper case letters to represent time series observations, as they are considered to be random variables, whereas in Chapter 1 we were treating them primarily as sets of numbers.

In this section we state a series of theorems summarizing the sampling properties (mean, variance, and asymptotic distribution) of \bar{X} , \bar{R} , $\hat{\rho}$, and \hat{f} . We will investigate the sampling properties of the sample partial autocorrelation function in Section 3.4.4 (see Theorem 3.4.6).

Since the proofs of the theorems in this section are rather lengthy and widely available elsewhere (see Chapter 8 of Anderson (1971) or Chapter 5 of Priestley (1981), for example), we will not provide them here. Rather, we attempt to give summaries of the implications of the theorems on analyzing data.

3.1.1. The Sample Mean

Theorem 3.1.1 SAMPLING PROPERTIES OF \bar{X}

Let X be a covariance stationary time series with mean μ and autocovariance function R . Let $\mathbf{X}_n = (X(1), \dots, X(n))^T$ be a realization of length n from X and let $\bar{X}_n = \frac{1}{n} \sum_{t=1}^n X(t)$. Then

a) $E(\bar{X}_n) = \mu$.

b) $\text{Var}(\bar{X}_n) \rightarrow 0$ as $n \rightarrow \infty$ if and only if $\text{Cov}(X(n), \bar{X}_n) \rightarrow 0$ as $n \rightarrow \infty$, a sufficient condition for which is that $R(n) \rightarrow 0$ as $n \rightarrow \infty$.

c) If $\sum_{v=-\infty}^{\infty} R(v) < \infty$, then

$$\lim_{n \rightarrow \infty} n \text{Var}(\bar{X}_n) = \sum_{v=-\infty}^{\infty} R(v),$$

which is equal to $f(0)$ if X has a spectral density f and f is continuous at frequency zero, a sufficient condition for which is that R is absolutely summable.

d) If X is a general linear process with independent, identically distributed errors and absolutely summable coefficients, then

$$\sqrt{n}(\bar{X}_n - \mu) \xrightarrow{\mathcal{L}} N(0, f(0)).$$

Implications: Part (a) says that \bar{X}_n is an unbiased estimator of μ . A sufficient condition for an estimator to be consistent is that it is asymptotically unbiased

(that is, has expectation converging to the parameter being estimated) and has variance converging to zero. Thus part (b) gives a condition for \bar{X}_n to be consistent. A time series is also said to be ergodic in the mean if $\text{Var}(\bar{X}_n) \rightarrow 0$ as $n \rightarrow \infty$, that is, if the average \bar{X}_n over time of a single realization approaches the average $\mu = E(X(t))$ at a single time t of the ensemble of all possible realizations. Thus X is ergodic in the mean if and only if as n gets large, adding new observations in the calculation of \bar{X}_n has no effect on it in the sense that $\text{Cov}(X(n), \bar{X}_n) \rightarrow 0$.

Part (c) gives two expressions that can be used to approximate $\text{Var}(\bar{X}_n)$ and also shows that $\text{Var}(\bar{X}_n)$ goes to zero at the rate of $1/n$. Finally, part (d) provides a general Central Limit Theorem for \bar{X}_n . This gives

$$\bar{X}_n \pm Z_{\alpha/2} \sqrt{\frac{f(0)}{n}}$$

as a $100(1-\alpha)\%$ confidence interval for μ . We will see in Section 3.3 that under the conditions of part (d), we can find an estimator $\tilde{f}(0)$ such that

$$\bar{X}_n \pm Z_{\alpha/2} \sqrt{\frac{\tilde{f}(0)}{n}}$$

is also a $100(1-\alpha)\%$ large sample confidence interval for μ .

Equivalent Number of Uncorrelated Observations

To illustrate the results of Theorem 3.1.1, suppose that X is an AR(1) process with coefficient α , noise variance σ^2 , and mean μ . Then

$$f(0) = \sigma^2 \frac{1}{|1 + \alpha e^{2\pi i 0}|^2} = \frac{\sigma^2}{(1 + \alpha)^2},$$

which we can write in terms of $\rho = \rho(1) = -\alpha$ as

$$f(0) = \frac{R(0)(1 - \rho^2)}{(1 - \rho)^2} = R(0) \frac{1 + \rho}{1 - \rho},$$

since $\sigma^2 = R(0)(1 - \rho^2)$. Thus a 95% confidence interval for μ is given by

$$\bar{X}_n \pm 1.96 \sqrt{\frac{R(0)}{n} \frac{1 + \rho}{1 - \rho}}.$$

This expression allows us to introduce the idea of an equivalent number of uncorrelated observations. Recall that a 95% confidence interval for the mean of a population having variance $R(0)$ based on a random sample of size N is given by

$$\bar{X} \pm 1.96 \sqrt{\frac{R(0)}{N}}.$$

Thus to make sure that the confidence interval from the AR(1) process and the random sample have the same width, we would need

$$N = n \frac{1 - \rho}{1 + \rho}.$$

For example, a sample of size $n = 100$ from an AR(1) process having $\rho = .8$ is equivalent in this sense to a random sample of size 11 from a population having the same variance as that of the AR(1) process. On the other hand, if ρ is negative, there is actually more information about μ in an AR(1) realization than in a random sample of the same size. In Figure 3.1 we illustrate the effect of ignoring autocorrelation in the calculation of confidence intervals for the mean of an autoregressive process. To obtain this figure, we generated 100 realizations of length 100 for the process having $\alpha = -.5$ (the macro used to create these graphs is described in Example 3.1). The confidence intervals which do not take autocorrelation into account fail to cover the true value of μ (which is 0) far more often than could reasonably be expected.

In general, a realization of length n from a time series having spectral density f is equivalent for estimating μ to a random sample of size N from a population having the same variance $R(0)$ as the time series, where

$$N = \frac{nR(0)}{f(0)} = \frac{n}{\sum_{v=-\infty}^{\infty} \rho(v)}.$$

Thus if $f(0)$ is large (small), estimation of μ for the time series case is less (more) accurate than for the corresponding random sample case (see Problem C3.1).

3.1.2. The Sample Autocovariances and Autocorrelations

In Chapter 1 we stated that the sample autocovariance function

$$\hat{R}(v) = \frac{1}{n} \sum_{t=1}^{n-|v|} (X(t) - \bar{X})(X(t+|v|) - \bar{X}), \quad |v| < n,$$

is generally accepted as the most satisfactory estimator of the autocovariance function R . The most popular alternative to \hat{R} is the “unbiased” estimator

$$\begin{aligned} \tilde{R}(v) &= \frac{1}{n-|v|} \sum_{t=1}^{n-|v|} (X(t) - \bar{X})(X(t+v) - \bar{X}) \\ &= \frac{n}{n-|v|} \hat{R}(v), \end{aligned}$$

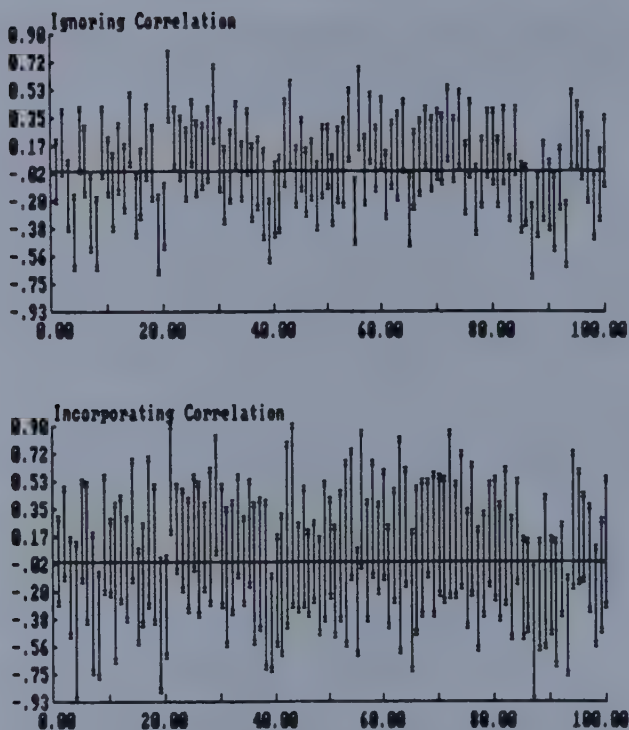


Figure 3.1. Confidence Intervals for $\mu = 0$ for an $AR(1)$.

so called because if we knew μ and used it instead of \bar{X} in the definition of \hat{R} , the result would be unbiased, whereas $\hat{R}(v)$ with μ replacing \bar{X} would have bias $-|v|/n$. However, we rarely know μ so neither \hat{R} nor \tilde{R} is unbiased, and in fact their bias is small relative to their variability. Further, it is generally believed (see Example 3.2 and Priestley (1981), p. 324) that the mean square error of $\hat{R}(v)$ is smaller than that of $\tilde{R}(v)$.

The second reason for preferring \hat{R} to \tilde{R} is that if we define $\hat{R}(v) = 0$ and $\tilde{R}(v) = 0$ for $|v| \geq n$, we have that \hat{R} is a positive definite sequence whereas \tilde{R} need not be. This is an algebraic fact which we prove in the next theorem. For an example of where \tilde{R} is not positive definite, see Problem T3.6.

Theorem 3.1.2 POSITIVE DEFINITENESS OF \hat{R}

Let $X(1), \dots, X(n)$ be a sequence of numbers and

$$\hat{R}(v) = \begin{cases} \frac{1}{n} \sum_{t=1}^{n-|v|} (X(t) - \bar{X})(X(t+|v|) - \bar{X}), & |v| < n \\ 0, & |v| \geq n, \end{cases}$$

where $\bar{X} = \frac{1}{n} \sum_{t=1}^n X(t)$. Then the sequence \hat{R} is positive definite if and only if $X(1), \dots, X(n)$ are not all the same.

Proof: We need to show that

$$\hat{\Gamma}_M = \text{Toepl}(\hat{R}(0), \dots, \hat{R}(M-1))$$

is positive definite for all $M \geq 1$ if and only if the X 's are not all the same. Let

$$Z(t) = \begin{cases} X(t) - \bar{X}, & t = 1, \dots, n \\ 0, & t > n. \end{cases}$$

Then the Z 's are not all zero if and only if the X 's are not all the same. Now it is easy to see that

$$n\hat{\Gamma}_M = \mathbf{Z}_M^T \mathbf{Z}_M,$$

where \mathbf{Z}_M is the $(n+M) \times M$ matrix having first column $\mathbf{z}_M = (Z(1), \dots, Z(n+M))^T$ with each successive column being the one before it shifted down one element with a zero placed on top. Thus $\hat{\Gamma}_M$ is positive definite if and only if

$$\mathbf{h}^T \mathbf{Z}_M^T \mathbf{Z}_M \mathbf{h} = (\mathbf{Z}_M \mathbf{h})^T (\mathbf{Z}_M \mathbf{h}) = \mathbf{w}_M^T \mathbf{w}_M > 0$$

for any nonzero $(n+M)$ -dimensional vector \mathbf{h} . But $\mathbf{w}_M^T \mathbf{w}_M > 0$ if and only if $\mathbf{w}_M = \mathbf{Z}_M \mathbf{h} \neq 0$, that is, if and only if \mathbf{Z}_M is of full rank. Now let $Z(j)$ be the first nonzero Z (which exists if and only if the X 's are not all the same). Then the j th through $(M+j-1)$ st rows of \mathbf{Z}_M form a lower triangular $(M \times M)$ matrix having $Z(j)$ on the diagonal. This matrix has nonzero determinant and thus these rows are linearly independent and \mathbf{Z}_M is of full rank.

In the rest of this section we will assume that μ is known and has been subtracted from the data so that the time series from which we are sampling has mean zero. Thus we consider the properties of

$$\hat{R}(v) = \frac{1}{n} \sum_{t=1}^{n-|v|} X(t)X(t+|v|), \quad |v| < n,$$

and note that these properties are a good approximation to the properties of the more realistic case where we must subtract \bar{X} (see Anderson (1971), p. 471). In finding the variance of $\hat{R}(v)$ we need to find $E(\hat{R}^2(v))$ which involves finding the expectation of the product of four X 's.

Definition. A covariance stationary time series is called fourth-order stationary if

$$E(X(t)X(t+h)X(t+r)X(t+s))$$

is a function $\gamma(h, r, s)$ of h, r , and s only. The function

$$\kappa(h, r, s) = \gamma(h, r, s) - R(h)R(r-s) - R(r)R(h-s) - R(s)R(h-r)$$

is called the fourth order cumulant function of X .

If X is a Gaussian process, then

$$\gamma(h, r, s) = R(h)R(r-s) + R(r)R(h-s) + R(s)R(h-r)$$

(Isserlis (1918)), and so $\kappa(h, r, s) = 0$. Some of the results in this section will have expressions including these cumulants but in practice we usually assume that they are zero, which they in fact are for a Gaussian process.

Theorem 3.1.3 PROPERTIES OF \hat{R} AND $\hat{\rho}$

Let X be a zero mean, fourth-order stationary time series having fourth order cumulant function κ , and let \hat{R} and $\hat{\rho}$ be the sample autocovariance and autocorrelation functions of a realization of length n from X . Then

$$\text{a) } E(\hat{R}(v) - R(v)) = -\frac{|v|}{n}R(v) \rightarrow 0 \text{ as } n \rightarrow \infty.$$

b) If all of the series involved converge, then

$$\begin{aligned} \lim_{n \rightarrow \infty} n \text{Cov}(\hat{R}(h), \hat{R}(g)) &= \sum_{r=-\infty}^{\infty} [R(r)R(r+h-g) + R(r-g)R(r+h)] \\ &\quad + \sum_{r=-\infty}^{\infty} \kappa(h, -r, g-r). \end{aligned}$$

If X has a continuous spectral density f , then

$$\begin{aligned} \lim_{n \rightarrow \infty} n \text{Cov}(\hat{R}(h), \hat{R}(g)) &= 2 \int_0^1 \cos 2\pi h\omega \cos 2\pi g\omega f^2(\omega) d\omega \\ &\quad + \sum_{r=-\infty}^{\infty} \kappa(h, -r, g-r) \end{aligned}$$

if the series converges. If X is a general linear process with absolutely summable coefficients and independent errors having variance σ^2 and finite fourth cumulant λ_4 , then

$$\sum_{r=-\infty}^{\infty} \kappa(h, -r, g - r) = \frac{\lambda_4}{\sigma^4} R(h)R(g).$$

c) If X is a general linear process with absolutely summable coefficients and independent errors, then

$$\begin{aligned} \lim_{n \rightarrow \infty} n \text{Cov}(\hat{\rho}(h), \hat{\rho}(g)) &= \sum_{r=-\infty}^{\infty} \left[\rho(r+g)\rho(r+h) + \rho(r-g)\rho(r+h) - 2\rho(h)\rho(r)\rho(r+g) \right. \\ &\quad \left. - 2\rho(g)\rho(r)\rho(r+h) + 2\rho(g)\rho(h)\rho^2(r) \right] \\ &= \frac{2}{R^2(0)} \int_0^1 [\cos 2\pi h\omega - \rho(h)][\cos 2\pi g\omega - \rho(g)] f^2(\omega) d\omega. \end{aligned}$$

d) If X is a general linear process with absolutely summable coefficients and independent errors having variance σ^2 and finite fourth cumulant λ_4 , then any finite collection of \hat{R} 's or $\hat{\rho}$'s is asymptotically multivariate normal with asymptotic covariances given in parts (b) and (c).

Implications: Part (a) says that the bias in $\hat{R}(v)$ is of the order $1/n$; that is, n times the bias converges to a constant. For fixed n , $|v|/n$ is increasing with $|v|$, but $R(v)$ should be decreasing with v so that the net effect should be that the bias is negligible as n increases. Note again if X is a Gaussian process that the cumulant terms in parts (b)–(d) will vanish. Note also that they do in part (c) for the asymptotic covariances of the sample autocorrelation function even if the process is not Gaussian. Parts (b) and (c) show the important fact that the elements of the covariance or correlation function are themselves correlated, which means that inferences made about them for different lags can not be done independently. The first expression in part (c) is called Bartlett's formula, while the frequency domain formula in part (c) is due to Parzen (1961). We can get confidence intervals for $R(v)$ or $\rho(v)$ (see Example 3.3) from parts (b) and (c) since

$$\begin{aligned} \lim_{n \rightarrow \infty} n \text{Var}(\hat{R}(v)) &= \sum_{r=-\infty}^{\infty} [R^2(r) + R(r-v)R(r+v)] \\ &= 2 \int_0^1 \cos^2 2\pi v\omega f^2(\omega) d\omega, \end{aligned}$$

while

$$\begin{aligned}
 \lim_{n \rightarrow \infty} n \text{Var}(\hat{\rho}(v)) &= \sum_{r=-\infty}^{\infty} \left[\rho^2(r) + \rho(r-v)\rho(r+v) \right. \\
 &\quad \left. - 4\rho(v)\rho(r)\rho(r+v) + 2\rho^2(v)\rho^2(r) \right] \\
 &= \frac{2}{R^2(0)} \int_0^1 [\cos 2\pi v\omega - \rho(v)]^2 f^2(\omega) d\omega.
 \end{aligned}$$

If X is a white noise series with variance $R(0) = \sigma^2$, we have, approximately for large samples, that the elements of the covariance and correlation sequences are uncorrelated, and for $v > 0$,

$$\begin{aligned}
 \hat{R}(v) &\sim N\left(0, \frac{\sigma^4}{n}\right) \\
 \hat{\rho}(v) &\sim N\left(0, \frac{1}{n}\right),
 \end{aligned}$$

while $\hat{R}(0) \sim N(\sigma^2, 2\sigma^4/n)$.

3.1.3. The Sample Spectral Density Function

As in Section 3.1.2, we will assume that the mean of the process being considered is zero. Thus we consider the properties of

$$\begin{aligned}
 \hat{f}(\omega) &= \frac{1}{n} \left| \sum_{t=1}^n X(t) e^{2\pi i(t-1)\omega} \right|^2, \quad \omega \in [0, 1] \\
 &= \sum_{v=-(n-1)}^{n-1} \hat{R}(v) e^{-2\pi i v \omega}.
 \end{aligned}$$

Anderson (1971), p. 476 gives a heuristic argument that the properties of \hat{f} with and without subtracting \bar{X} are essentially the same. The next theorem verifies our observation in Chapter 1 that the sample spectral density function is too oscillatory to be useful for making statistical inferences.

Theorem 3.1.4

PROPERTIES OF THE PERIODOGRAM

Let X be a zero mean, fourth-order stationary time series having fourth-order cumulant function κ and let \hat{f} be the sample spectral density function for a realization of length n from X . Then

a) If X has spectral density function f , then

$$\begin{aligned} E(\hat{f}(\omega)) &= \sum_{v=-(n-1)}^{n-1} \left(1 - \frac{|v|}{n}\right) R(v) \cos 2\pi v\omega \\ &= \int_0^1 F_n(\omega - \tau) f(\tau) d\tau, \end{aligned}$$

where F_n is the Féjer kernel

$$F_n(\omega) = \sum_{v=-n}^n \left(1 - \frac{|v|}{n}\right) \cos 2\pi v\omega = \frac{1}{n} \left(\frac{\sin \pi n\omega}{\sin \pi \omega} \right)^2.$$

b) If $\sum_{v=-\infty}^{\infty} R(v) \cos 2\pi v\omega < \infty$, then

$$\lim_{n \rightarrow \infty} E(\hat{f}(\omega)) = \sum_{v=-\infty}^{\infty} R(v) \cos 2\pi v\omega,$$

which is equal to $f(\omega)$ if X has a spectral density f that is continuous at frequency ω .

c) If $\sum_{v=-\infty}^{\infty} |v R(v)| < \infty$, then R is absolutely summable and

$$\lim_{n \rightarrow \infty} n \left[E(\hat{f}(\omega)) - f(\omega) \right] = -2 \sum_{v=1}^{\infty} v R(v) \cos 2\pi v\omega.$$

d) If X has a spectral density that is continuous at frequency ω , and

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{r, u, w=-(n-1)}^{n-1} |\kappa(r, u, w)| = 0,$$

then

$$\lim_{n \rightarrow \infty} \text{Var}(\hat{f}(\omega)) = \begin{cases} f^2(\omega), & \omega \neq 0, .5 \\ 2f^2(\omega), & \omega = 0, .5, \end{cases}$$

while if f is bounded in $(\omega_1 - \delta, \omega_1 + \delta)$ and $(\omega_2 - \delta, \omega_2 + \delta)$ for some $\delta > 0$, we have

$$\lim_{n \rightarrow \infty} \text{Cov}(\hat{f}(\omega_1), \hat{f}(\omega_2)) = 0, \quad \omega_1 \neq \omega_2.$$

e) If X is a general linear process having absolutely summable coefficients and independent, identically distributed errors, then for any integer $M \geq 1$ and fixed frequencies $\omega_1, \dots, \omega_M$ in $(0, .5)$, we have that the random variables

$$\frac{2\hat{f}(\omega_1)}{f(\omega_1)}, \dots, \frac{2\hat{f}(\omega_M)}{f(\omega_M)}$$

converge in distribution to the M independent random variables Q_1, \dots, Q_M , each having the chi-square distribution with two degrees of freedom. Also the result continues to hold at all of the natural frequencies, that is, for the frequencies of the form $\omega_j = (j-1)/n$. The asymptotic distributions of $\hat{f}(0)/f(0)$ and $\hat{f}(.5)/f(.5)$ are chi-square with one degree of freedom.

f) If X is a normal white noise process with variance σ^2 , then the periodogram ordinates, that is, the values of the sample spectral density at the natural frequencies, are independently distributed and

$$\frac{2\hat{f}(\omega_j)}{\sigma^2} \sim \chi_2^2, \quad \omega_j \neq 0, .5$$

$$\frac{\hat{f}(\omega_j)}{\sigma^2} \sim \chi_1^2, \quad \omega_j = 0, .5.$$

Implications: Parts (a) and (b) show that the sample spectral density is asymptotically unbiased under general conditions, while part (c) shows that the bias is of order n . More importantly, these parts show that \hat{f} is actually estimating an integrated weighted average of the values of f in a neighborhood of the frequency ω . In Figure 3.2 we show the graph of the Féjer kernel F_n for several values of n . This weight function is converging with n to a delta function, and thus unless the true spectral density has a very sharp peak in the neighborhood of ω , the integral will converge to $f(\omega)$. See Example 3.5 for the macro that was used to produce this figure.

Part (d) shows that the sample spectral density function is not consistent since its variance does not go to zero. Further, the variance at frequency ω is a constant independent of sample length. We illustrate this in Figure 3.3 (see Example 3.4) where we have given the log periodogram of realizations of lengths 200, 400, 600, and 800 for a normal white noise series having variance 1. Notice that the variability in these graphs is not decreasing with sample length.

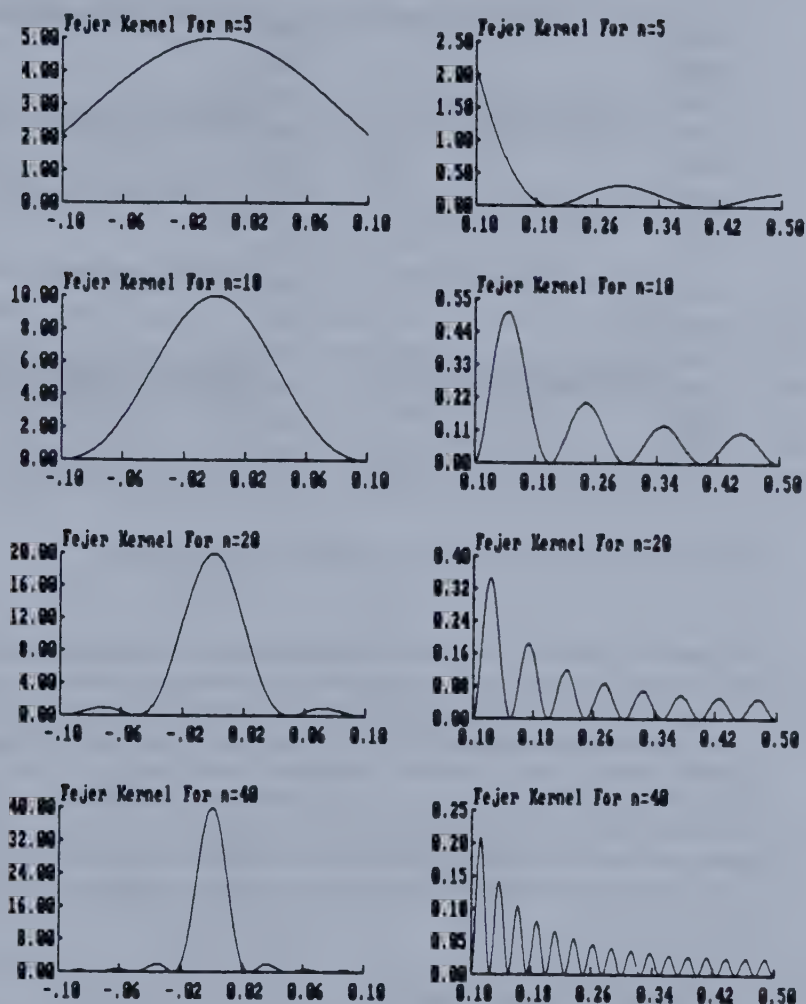


Figure 3.2. The Féjer Kernel for Various Values of n .

The fact that the variance of $\hat{f}(\omega)$ is a function of $f^2(\omega)$, that is, the square of the mean of $\hat{f}(\omega)$, is another reason for plotting the log of the periodogram instead of the periodogram itself, as it is easy to show (see Problem T3.3) that the limiting variance of the log of $\hat{f}(\omega)$ is a constant independent of ω .

Part (d) also shows that the values of the sample spectral density are asymptotically uncorrelated at different frequencies. Thus the log periodogram

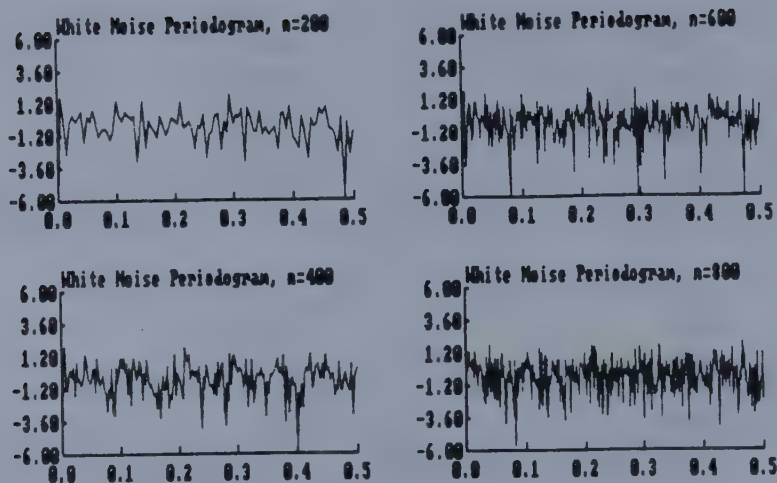


Figure 3.3. Illustrating the Inconsistency of the Periodogram.

appears to be a white noise series itself except that its level will change with the level of the true spectral density. We will see in the next section that we can make use of this uncorrelatedness to produce smooth versions of the sample spectral density that will in fact be consistent estimates of the spectral density. Parts (e) and (f) provide information on the asymptotic distribution of the sample spectral density. In essence, the sample spectral density has the chi-square distribution for large n in general, while if X is a Gaussian white noise series, then the periodogram is exactly chi-square for all n (see Problem T3.5). We will use this fact in the next section when we consider tests for white noise.

3.2. Tests for White Noise

In this section we consider two widely used methods for testing whether it is reasonable to conclude that a data set is a realization from a white noise time series.

3.2.1. Bartlett's Test

Suppose that $X(1), \dots, X(n)$ is a realization from a $WN(\sigma^2)$ process. Thus the spectral distribution function F of X is given by $F(\omega) = \sigma^2\omega$ for $\omega \in [0, 1]$,

and we would expect that the cumulative periodogram

$$\hat{F}(\omega_k) = \frac{\sum_{j=1}^k \hat{f}(\omega_j)}{\sum_{j=1}^q \hat{f}(\omega_j)}, \quad k = 1, \dots, q = [n/2] + 1,$$

of the data should be close to the points

$$S_k = \frac{k}{q}, \quad k = 1, \dots, q.$$

Recall from Theorem 3.1.4 that except for $\omega_j = 0$ and $.5$, the random variables $2\hat{f}(\omega_j)/\sigma^2$ are asymptotically independent and identically distributed as χ_2^2 random variables. But a χ_2^2 distributed random variable is the same as 2 times a variable having the exponential distribution with mean 1, and thus it can be shown that $\hat{F}(\omega_1), \dots, \hat{F}(\omega_q)$ have approximately the same distribution as the ordered values $U_1 \leq \dots \leq U_q$ of a sample of size q from a $U(0,1)$ population. Thus we can use the well-known result (see Feller (1948), for example) that

$$\lim_{q \rightarrow \infty} \Pr \left(\max_{1 \leq k \leq q} \sqrt{q} \left| U_k - \frac{k}{q} \right| \leq a \right) = \sum_{j=-\infty}^{\infty} (-1)^j e^{-2a^2 j^2} \equiv G(a)$$

to judge whether a cumulative periodogram has a maximum deviation from the expected straight line too extreme to be reasonable under the hypothesis of white noise. This result is used in the **WNTEST** macro in Chapter 1. See Bartlett (1955), pp. 92–94 for more details about this procedure.

The **BARTTEST** Command

Given integers **n** and **Q**, and an array **F** of length $q = [Q/2] + 1$, the command

pval=BARTTEST(F,Q,n,B)

will return

$$B = \max_{1 \leq k \leq q} \sqrt{\frac{n}{2}} \left| F(k) - \frac{k}{q} \right|$$

$$p = 1 - G(B)$$

in the real scalar variables **B** and **pval**, respectively. The command

pval=BARTTEST(B)

will return **pval** = $1 - G(B)$ for a user-specified positive value **B**. In Figure 3.4 we display the cdf $F(b) = \Pr(B \leq b)$, while in Table 3.1 we list the values of

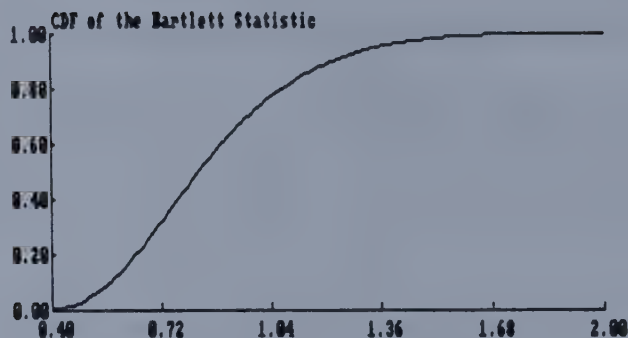


Figure 3.4. Cumulative Distribution Function for the Bartlett Statistic.

$F(b)$ for $b = .40, .41, \dots, 2.00$. The macro that generated the figure and table is discussed in Example 3.6.

3.2.2. The Portmanteau Test

Bartlett's test is done in the frequency domain and uses the fact that the spectral density of white noise is constant. Another approach to the problem of testing for white noise is to use the fact that its autocorrelation coefficients for nonzero lag are all zero. The most popular method of this type is often called the portmanteau or Q test (see Box and Pierce (1970) and Ljung and Box (1978)). If $X(1), \dots, X(n)$ is a sample realization from a white noise process, then the statistic

$$Q = n(n+2) \sum_{j=1}^m \frac{1}{n-j} \hat{\rho}^2(j) \xrightarrow{L} \chi_m^2.$$

Thus the hypothesis of white noise is rejected if $Q > \chi_{\alpha, m}^2$. Note that this test is often applied to the residuals (one step ahead forecast errors) from a model that has been fit to a data set. In this case, one degree of freedom is subtracted from m for each parameter that has been estimated.

The QTEST Command

The command

```
pval=QTEST(rho,m,p,q,n,Q)
```

will return the value Q of the test statistic and its p -value (in `pval`) for the errors of an $\text{ARMA}(p,q)$ fit to a time series of length n . If `QTEST` is used for an original time series, then p and q should be entered as zero.

Table 3.1. Values of the CDF for the Bartlett Statistic

b	F(b)	b	F(b)	b	F(b)	b	F(b)
.400	.003	.410	.004	.420	.005	.430	.007
.440	.010	.450	.013	.460	.016	.470	.020
.480	.025	.490	.030	.500	.036	.510	.043
.520	.050	.530	.059	.540	.067	.550	.077
.560	.088	.570	.099	.580	.110	.590	.123
.600	.136	.610	.149	.620	.163	.630	.178
.640	.193	.650	.208	.660	.224	.670	.240
.680	.256	.690	.272	.700	.289	.710	.305
.720	.322	.730	.339	.740	.356	.750	.373
.760	.390	.770	.406	.780	.423	.790	.440
.800	.456	.810	.472	.820	.488	.830	.504
.840	.519	.850	.535	.860	.550	.870	.565
.880	.579	.890	.593	.900	.607	.910	.621
.920	.634	.930	.647	.940	.660	.950	.673
.960	.685	.970	.696	.980	.708	.990	.719
1.000	.730	1.010	.741	1.020	.751	1.030	.761
1.040	.770	1.050	.780	1.060	.789	1.070	.798
1.080	.806	1.090	.814	1.100	.822	1.110	.830
1.120	.837	1.130	.845	1.140	.851	1.150	.858
1.160	.864	1.170	.871	1.180	.877	1.190	.882
1.200	.888	1.210	.893	1.220	.898	1.230	.903
1.240	.908	1.250	.912	1.260	.916	1.270	.921
1.280	.925	1.290	.928	1.300	.932	1.310	.935
1.320	.939	1.330	.942	1.340	.945	1.350	.948
1.360	.951	1.370	.953	1.380	.956	1.390	.958
1.400	.960	1.410	.962	1.420	.965	1.430	.967
1.440	.968	1.450	.970	1.460	.972	1.470	.973
1.480	.975	1.490	.976	1.500	.978	1.510	.979
1.520	.980	1.530	.981	1.540	.983	1.550	.984
1.560	.985	1.570	.986	1.580	.986	1.590	.987
1.600	.988	1.610	.989	1.620	.989	1.630	.990
1.640	.991	1.650	.991	1.660	.992	1.670	.992
1.680	.993	1.690	.993	1.700	.994	1.710	.994
1.720	.995	1.730	.995	1.740	.995	1.750	.996
1.760	.996	1.770	.996	1.780	.996	1.790	.997
1.800	.997	1.810	.997	1.820	.997	1.830	.998
1.840	.998	1.850	.998	1.860	.998	1.870	.998
1.880	.998	1.890	.998	1.900	.999	1.910	.999
1.920	.999	1.930	.999	1.940	.999	1.950	.999
1.960	.999	1.970	.999	1.980	.999	1.990	.999
2.000	.999						

The Choice of m in the Q Test

One difficulty in using the `QTEST` command is in deciding what value of m to use. Asymptotically, this choice shouldn't matter, but to investigate its effect in small and moderate samples, we give in Table 3.2 the results of a simple simulation study. For samples of size 50, 100, and 200, we generated 500 Gaussian white noise series and counted how many times the null hypothesis of white noise was rejected for $\alpha = .01, .05$, and $.10$ for $m=5, 10$, and 20 . For each sample size, these should be 5, 25, and 50 for the respective values of α . For each α , we also counted the number of times out of 500 that the conclusion was not the same for the three values of m . From these results, it appears that (1) the true Type I error probability of the Q test tends to increase with m , particularly for smaller sample sizes, and (2) the effect of choosing m is more important as α increases. The reader may want to replicate this experiment to see if these patterns persist. The macros that were used to find the results are given in Example 3.7. Note that the `QTEST` and `BARTTEST` commands can be used in conjunction with the `WNTTEST` macro discussed in Example 1.5.

Table 3.2. Results of a Simulation Study of the Effect of the Choice of m in the Q Test

	alpha=.01			alpha=.05			alpha=.10		
	n=50	n=100	n=200	n=50	n=100	n=200	n=50	n=100	n=200
m=5	8	8	6	31	27	27	46	52	35
m=10	14	6	4	31	26	25	55	56	54
m=20	18	14	6	46	43	23	72	70	53
# diff.	21	20	11	57	48	45	81	82	74

3.3. Nonparametric Spectral Density Estimation

We saw in Section 3.1.3 that the sample spectral density function \hat{f} is an inconsistent estimator of the spectral density function f . In this section we describe how \hat{f} can be modified to produce consistent estimators. In Section 3.8 we will use ARMA models to obtain other consistent estimators, but for now we will not assume that X follows any particular model. Thus we call the resulting estimation procedures nonparametric in analogy with nonparametric statistics where no parametric assumptions are made about the distribution of a data set. We will see in Section 3.8 that if the true spectral density being estimated is that of an invertible ARMA model, then using parametric methods will provide spectral estimates having better properties than the nonparametric methods.

3.3.1. Smoothing the Sample Spectral Density

The basic problem with \hat{f} is that it is too wiggly to be an adequate estimator of a function that is typically smooth over much of its domain. In this section we consider a simple averaging approach to smoothing the periodogram. In Section 3.3.2 we will consider more elaborate smoothing methods. We will present the simple averaging in some detail as it illustrates the general ideas of smoothing and is not as difficult mathematically as the methods in Section 3.3.2.

Recall from part (e) of Theorem 3.1.4 that under general conditions

$$\hat{f}(0), \hat{f}\left(\frac{1}{n}\right), \dots, \hat{f}\left(\frac{[n/2]}{n}\right)$$

are asymptotically independent random variables with

$$\hat{f}(\omega_j) \sim \begin{cases} \frac{f(\omega_j)}{2} \chi_2^2, & \omega_j \neq 0, .5 \\ f(\omega_j) \chi_1^2, & \omega_j = 0, .5. \end{cases}$$

Consider estimating f at one of the natural frequencies $\omega_j = (j-1)/n$ by

$$\tilde{f}(\omega_j) = \frac{1}{2m+1} \sum_{k=-m}^m \hat{f}(\omega_{j+k}),$$

that is, by the average of $\hat{f}(\omega_j)$ and the m values of the periodogram on either side of it. If $\omega_{j-m} < 0$ or $\omega_{j+m} > .5$, we can use the fact that \hat{f} is symmetric about 0 and .5 to obtain the elements in the sum.

In Example 3.8 we give a macro called **AVEPER** that can be used to do this simple averaging. To illustrate the procedure, consider Figure 3.5 which contains four graphs. The first is the log of the standardized form of the true spectral density of the AR(5) process:

$$\begin{aligned} X(t) + 1.7X(t-1) + 2.4X(t-2) + 1.634X(t-3) + 0.872X(t-4) \\ + 0.168X(t-5) = \epsilon(t), \end{aligned}$$

where ϵ is white noise with variance 1. We then give the averaged periodogram for a realization of length 200 for $m = 1, 4$, and 7. As m increases, the estimate becomes smoother (less variable), but the bias in the neighborhood of the peak becomes quite pronounced. This is to be expected as in this neighborhood the periodogram is changing rapidly. For the frequency where the peak occurs, the final estimate is the average of the periodogram at that frequency and the m

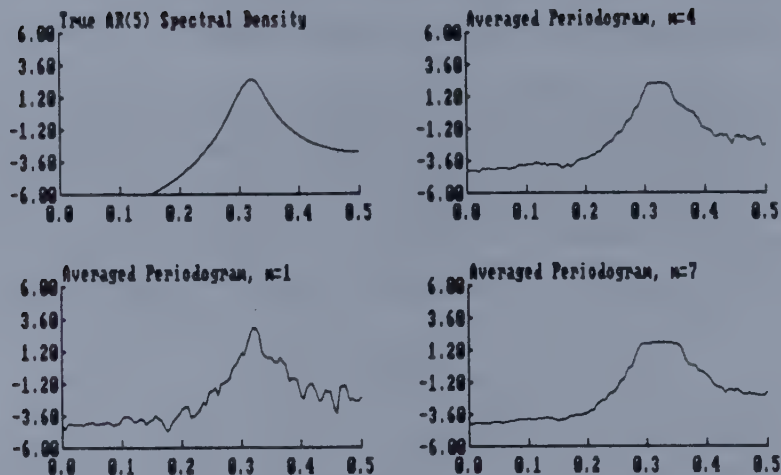


Figure 3.5. The Effect of Increasing m in Averaging the Periodogram.

frequencies on either side of it. Most of these values of the periodogram are much smaller than the value that is being estimated.

Now suppose that $\omega_{j-m} > 0$ and $\omega_{j+m} < .5$. We can think of $\tilde{f}(\omega_j)$ as averaging all of the periodogram values in the frequency band $\omega_j \pm m/n$, that is, as a smoother having bandwidth m/n . As in the kernel probability density estimator in Section 1.7, we consider bandwidth to be half the width of the frequency band, that is, the width of the interval on each side of the center frequency. Arguing heuristically, we assume that n is large and use the asymptotic properties of \tilde{f} to obtain

$$E(\tilde{f}(\omega_j)) \doteq \frac{1}{2m+1} \sum_{k=-m}^m f(\omega_{j+k})$$

$$\text{Var}(\tilde{f}(\omega_j)) \doteq \left(\frac{1}{2m+1} \right)^2 \sum_{k=-m}^m f^2(\omega_{j+k}) \leq \frac{1}{2m+1} \max_{-m \leq k \leq m} f^2(\omega_{j+k}).$$

These two expressions exhibit clearly the importance of the smoothness of f , as well as the tradeoff between variance and bias in $\tilde{f}(\omega_j)$ in the choice of m . For a fixed value of n , increasing m will increase the bandwidth and lead to including values of f at frequencies farther from the target frequency ω_j in the expected value of $\tilde{f}(\omega_j)$, thus increasing the possibility of a larger bias. If f is smooth in the neighborhood of ω_j , then this bias will not increase appreciably. On the other hand, increasing m has the effect of decreasing the variance of the estimator, again assuming that f does not significantly increase as frequencies

get farther from ω_j . Note that the simplest case is when X is white noise, in which case f is constant over $[0, .5]$, and we can increase the bandwidth without increasing the bias while decreasing the variance at the rate of $1/(2m+1)$. At the other extreme, we might be estimating f at a frequency that is adjacent to a band where f rises sharply to a peak, in which case increasing m could increase both the bias and the variance. This phenomenon is known as leakage as high power in f in one frequency band "leaks" into estimates of f at other frequencies.

As long as f is bounded, we can let m go to ∞ as n goes to ∞ so that the variance of $\tilde{f}(\omega_j)$ goes to zero. As m and n get large, we have

$$E(\tilde{f}(\omega_j)) \doteq \frac{1}{2m+1} \sum_{k=-m}^m f(\omega_{j+k}) \doteq \frac{n}{2m} \int_{\omega_j-m/n}^{\omega_j+m/n} f(\omega) d\omega.$$

If we let m/n get small and assume that f is such that it is roughly constant over a small interval, then the above integral is approximately $\frac{2m}{n} f(\omega_j)$, and $\tilde{f}(\omega_j)$ is asymptotically unbiased.

For two different fixed frequencies $\omega_j \neq \omega_k$, we have that $\tilde{f}(\omega_j)$ and $\tilde{f}(\omega_k)$ will become independent as m/n goes to zero since they will eventually be based upon averaging values of the periodogram for nonoverlapping frequency bands.

We next consider the asymptotic distribution of $\tilde{f}(\omega_j)$. For $\omega_{j-m} > 0$ and $\omega_{j+m} < .5$, we have

$$\tilde{f}(\omega_j) \approx \frac{1}{2m+1} \sum_{k=-m}^m \frac{f(\omega_{j+k})}{2} Q_{k+m+1},$$

where \approx means has the same distribution as, and the Q 's are independent χ^2_2 distributed random variables. We know that a pure sum of independent χ^2 random variables has itself a χ^2 distribution with degrees of freedom equal to the sum of the individual degrees of freedom. Thus if $X \sim \text{WN}(\sigma^2)$, we would have that $\tilde{f}(\omega_j) \approx \{\sigma^2/[2(2m+1)]\} \chi^2_{2(2m+1)}$. The distribution of a general linear combination of χ^2 random variables is often approximated by that of $Q = c\chi^2_\nu$, where c and ν are chosen so that the mean and variance of Q are the same as those of the linear combination (see Tukey (1949)). We will use this device to approximate the distribution of $\tilde{f}(\omega_j)/f(\omega_j)$. Thus we have

$$E(Q) = c\nu = E\left(\frac{\tilde{f}(\omega_j)}{f(\omega_j)}\right) \doteq 1$$

$$\text{Var}(Q) = 2c^2\nu = \text{Var}\left(\frac{\tilde{f}(\omega_j)}{f(\omega_j)}\right) \doteq \frac{1}{2(2m+1)}.$$

We then set $c\nu = 1$ and $2c^2\nu = 1/2(2m+1)$ and obtain $c = 1/\nu$ and $\nu = 2(2m+1)$, which gives

$$\frac{\nu \tilde{f}(\omega_j)}{f(\omega_j)} \sim \chi_\nu^2, \quad \omega_j \neq 0, .5.$$

Note that in this simple averaging case we could have obtained this immediately from our assumption that f is constant around ω_j .

Estimating $f(0)$ and $f(.5)$

The arguments given above must be modified when we consider estimating f at frequency 0 or .5, as then the values of \hat{f} on each side of the center frequency are the same. If we write, for example,

$$\tilde{f}(0) = \frac{1}{2m+1} \left[\hat{f}(0) + 2 \sum_{k=1}^m \hat{f}(\omega_{j+k}) \right],$$

we find

$$\frac{\nu \tilde{f}(\omega_j)}{2 f(\omega_j)} \sim \chi_{\nu/2}^2, \quad \omega_j = 0, .5.$$

3.3.2. Kernels and Fourier Series Approximations

The next natural step in estimating f is to insert a weight function into the averaging of the periodogram so that values of \hat{f} at frequencies far removed from ω_j get less weight than values at frequencies close to ω_j . We consider integrated weighted averages of \hat{f} of the form

$$\tilde{f}(\omega) = \int_0^1 K(\omega - \tau) \hat{f}(\tau) d\tau,$$

where K is a weight function called a (spectral) window. For a fixed frequency ω , this expression essentially says to superimpose the weight function onto the graph of the periodogram (with its value at zero centered at frequency ω), then multiply the two functions together (which down-weights values of \hat{f} far removed from the “center” frequency ω), and finally do the integral. To find the estimate at a different frequency, the same process is carried out with the weight function moved over to be centered at the new frequency. Thus the function K essentially dictates what part of \hat{f} can be “seen” when finding the estimate at a certain frequency. This is the origin of the term *window*. We will study windows of various shapes and discuss the effect of their width (see Figure 3.7 for some examples of windows).

Using spectral windows then is a natural extension of the idea of averaging the sample spectral density. We could use weighted averages that are sums

instead of integrals, but the integrated averages arise naturally from another point of view, which we now describe. Such averages arise naturally from considerations of the general theory of the Fourier series representation of a function on a finite interval (see Section A.2). An expression of the form

$$f(\omega) = \sum_{v=-\infty}^{\infty} R(v)e^{-2\pi i v \omega}$$

where

$$R(v) = \int_0^1 f(\omega) e^{2\pi i v \omega} d\omega$$

is called the Fourier series representation of f , and the coefficients $R(v)$ are called its Fourier coefficients. The periodogram

$$\hat{f}(\omega) = \sum_{v=-(n-1)}^{n-1} \hat{R}(v) e^{-2\pi i v \omega}$$

is thus actually an estimator of the n th partial sum

$$f_n(\omega) = \sum_{v=-(n-1)}^{n-1} R(v) e^{-2\pi i v \omega}.$$

From this it is clear that there are two sources of error in \hat{f} as an estimator of f : a truncation error due to using f_n to approximate f , and an estimation error due to having to estimate R . In the general theory of Fourier series it is well known that the approximation of some functions by their Fourier series can be improved by applying a sequence of weights $k_n(v)$ to R in f_n (see the bandpass filter example in Section 2.3, for example), giving

$$\begin{aligned} f_{n,k}(\omega) &= \sum_{v=-(n-1)}^{n-1} k_n(v) R(v) e^{-2\pi i v \omega} \\ &= \sum_{v=-(n-1)}^{n-1} k_n(v) \left(\int_0^1 f(\tau) e^{2\pi i v \tau} d\tau \right) e^{-2\pi i v \omega} \\ &= \int_0^1 f(\tau) \sum_{v=-(n-1)}^{n-1} k_n(v) e^{-2\pi i v(\omega - \tau)} d\tau \\ &= \int_0^1 f(\tau) K_n(\omega - \tau) d\tau. \end{aligned}$$

The sequence k_n is called a lag window, and the function

$$K_n(\omega) = \sum_{v=-(n-1)}^{n-1} k_n(v) e^{-2\pi i v \omega}, \quad \omega \in (-\infty, \infty),$$

is called a spectral window. Thus the integrated averages that we have been discussing arise naturally from the idea of applying weights to the sample autocovariances. The weights $k_n(v)$ are usually an even function of v and thus K_n is real and symmetric about .5. Further, $K_n(\omega)$ is defined for all $\omega \in (-\infty, \infty)$ and is periodic of period 1. This also means that K_n is symmetric about zero (see Problem T3.10).

Note that k_n and K_n are Fourier pairs since

$$\begin{aligned} \int_0^1 K_n(\omega) e^{2\pi i j \omega} d\omega &= \int_0^1 \left(\sum_{v=-(n-1)}^{n-1} k_n(v) e^{-2\pi i v \omega} \right) e^{2\pi i j \omega} d\omega \\ &= \sum_{v=-(n-1)}^{n-1} k_n(v) \int_0^1 e^{2\pi i (j-v) \omega} d\omega \\ &= k_n(j). \end{aligned}$$

This suggests the following definition.

Definition. A spectral estimator of the form

$$\begin{aligned} \hat{f}_{n,k}(\omega) &= \sum_{v=-(n-1)}^{n-1} k_n(v) \hat{R}(v) e^{-2\pi i v \omega} \\ &= \int_0^1 \hat{f}(\tau) K_n(\omega - \tau) d\tau \end{aligned}$$

is called a window estimator with lag window k_n and spectral window K_n . If $k_n(v) = 0$ for $|v| > M$ for some integer M (called a truncation point), we say that the estimator is of truncated form.

Note that the sample spectral density function itself is of the above form with $k_n(v)$ identically one, which gives that K_n is the Dirichlet kernel that we saw in Section 2.3 (see Figure 2.3). We will assume the following throughout this section:

1. $k_n(0) = 1$.

2. $\int_0^1 K_n^2(\omega) d\omega < \infty$.
3. For any $\epsilon > 0$, $K_n(\omega) \rightarrow 0$ uniformly as $n \rightarrow \infty$ for $|\omega| > \epsilon$.
4. The weights $k_n(v)$ decay to zero fast enough that

$$\lim_{n \rightarrow \infty} \frac{\sum_{v=-(n-1)}^{n-1} \frac{|v|}{n} k_n^2(v)}{\sum_{v=-(n-1)}^{n-1} k_n^2(v)} = 0.$$

Windows of Scale Parameter Form

A variety of weight functions have been suggested, most of the scale parameter form, that is,

$$k_n(v) = \lambda\left(\frac{v}{M}\right),$$

for some integer $M < n$ called the scale parameter, with λ being a function satisfying

1. $\lambda(0) = 1$.
2. $\lambda(-u) = \lambda(u)$.
3. $\int_{-\infty}^{\infty} \lambda^2(u) du < \infty$.

Definition. The function λ defined above is called a lag window generator, while the function

$$\Lambda(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \lambda(u) e^{iu\omega} du, \quad \omega \in (-\infty, \infty),$$

is called the spectral window generator corresponding to λ .

For lag windows of scale parameter form, it can be shown (see Parzen (1963a)) for large M that

$$K_n(\omega) \doteq 2\pi M \Lambda(2\pi M \omega).$$

In Table 3.3 we give information about eight of the lag and spectral window generators that are commonly used. The last five columns in the table con-

tain numerical quantities that are important for judging the adequacy of the window. We describe these quantities below.

The first five lag window generators are zero outside of $|u| \leq 1$ and thus $k_n(v) = 0$ for $|v| > M$ and $\hat{f}_{n,k}$ only involves values of $\hat{R}(v)$ for $|v| \leq M$, and M is then called a truncation point. This terminology is often used even when λ is not of truncated form. The last window was originally considered by Parzen (1958) and has been discussed by Cogburn and Davis (1974).

In Figure 3.6 (see Example 3.9), we display each of the spectral windows K_n for $n = 500$ and $M = 20$. For the first five windows, K_n is only a function of M . We plot each window in two parts, the first part being for frequencies between $-.1$ and $.1$, and the second part for frequencies between $.1$ and $.5$. The basic feature of the windows is that they decay rapidly from frequency zero and then rise back up again in what are called sidelobes. All of the windows converge to a delta function as M increases. This is analogous to letting the bandwidth of the simple averaging smoother of the previous section go to zero. The truncated periodogram, Tukey, and Parzen-Cogburn-Davis windows are negative in certain frequency bands, while the others are nonnegative. This is important because using a negative spectral window can result in a spectral estimator that is negative.

Since the main lobe of a window is not rectangular, it is difficult to measure how wide it is. In the last column of Table 3.3 we give what is called Parzen's measure of the bandwidth of a window, denoted B_P and defined to be the width of a rectangular window having the same value at $\omega = 0$ as K_n and the same area as K_n . Since the areas of all of the K_n are one, we have

$$B_P = \frac{1}{2\pi M \Lambda(0)} \doteq \frac{1}{K_n(0)}.$$

Notice how the relative appearance of the windows follows the values of B_P . For example, the Parzen window for $M = 20$ is wider than the Tukey window for $M = 20$.

In Figure 3.7, we have superimposed the Parzen and Tukey windows for truncation points 24 and 18, respectively. Note how the main lobes seem to line up, which could be predicted based on the bandwidths of the two windows.

The WINDOW Command

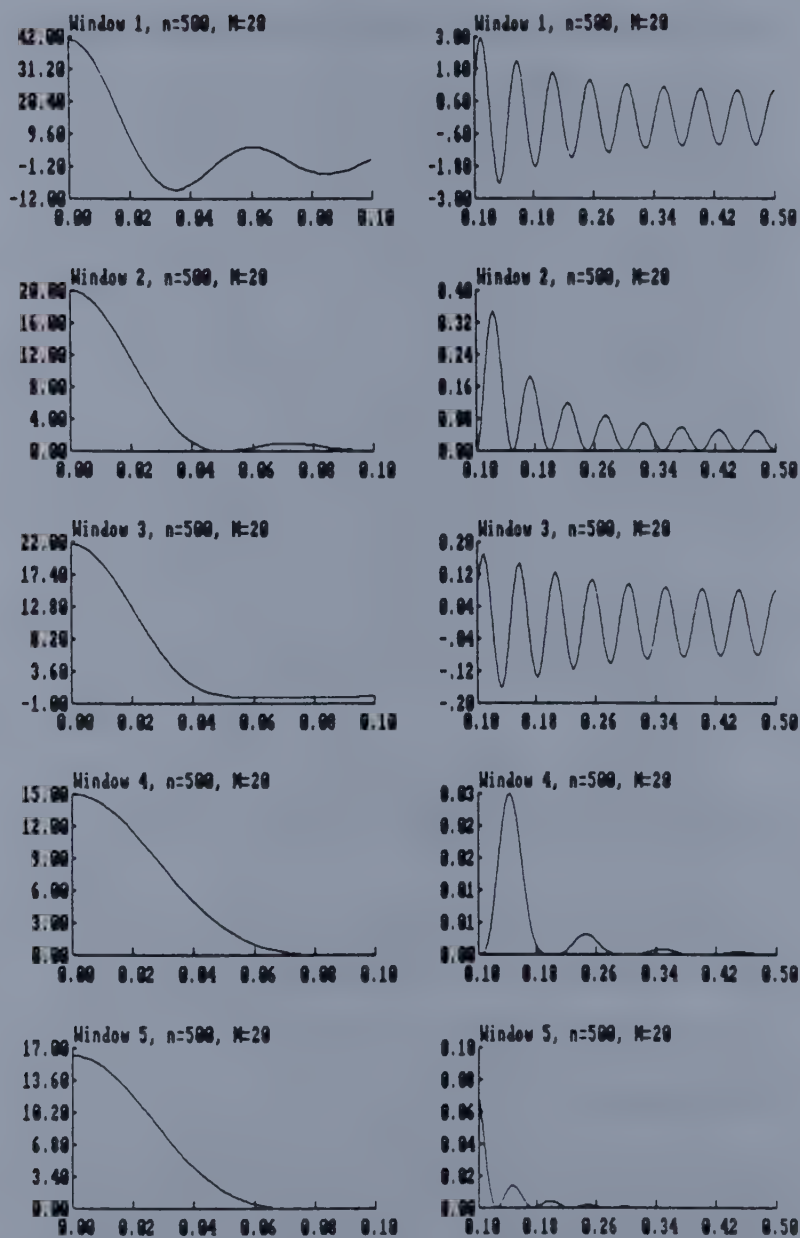
The command

```
f=WINDOW(rho,RO,M,Q,iopw[,n,c])
```

will calculate the nonparametric spectral density estimator at the Q frequencies $\omega_j = (j-1)/Q$ for $j = 1, \dots, Q$ and store the first $[Q/2] + 1$ of them in the array **f**. Input is **M** (the scale parameter to use), the sample variance **RO**, the number of frequencies **Q**, the integer **iopw** which determines which window is to be used (**iopw** is 1 through 8 corresponding to the eight windows in Table 3.3),

Table 3.3. Some Window Generators and Their Characteristics

$\Lambda(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \lambda(u)e^{i\omega u} du$	q	$\lambda(q)$	$\int_{-\infty}^{\infty} \lambda^2(u)du$	E_{min}	B_p
Generators of Truncated Form, i.e., $\lambda(u) = 0, u > 1$					
Truncated Periodogram Window: $\lambda(u) = 1$					
$\frac{1}{\pi} \frac{\sin \omega}{\omega}$	∞	-	2	-	$\frac{1}{2} \frac{1}{M}$
Bartlett Window: $\lambda(u) = 1 - u $					
$\frac{1}{2\pi} \left(\frac{\sin \omega/2}{\omega/2} \right)^2$	1	1	2/3	2/3	$\frac{1}{M}$
Tukey Window: $\lambda(u) = 0.54 + 0.46 \cos \pi u$					
$\frac{1}{2\pi} \frac{\sin \omega}{\omega} \frac{\pi^2}{\pi^2 - \omega^2}$	2	$0.23\pi^2$	0.795	1.20	$\frac{1}{M}$
Parzen Window: $\lambda(u) = \begin{cases} 1 - 6u^2 + 6 u ^3, & u \leq .5 \\ 2(1 - u)^3, & .5 \leq u \leq 1 \end{cases}$					
$\frac{3}{8\pi} \left(\frac{\sin \omega/4}{\omega/4} \right)^4$	2	6	0.539	1.32	$\frac{4}{3} \frac{1}{M}$
Bohman (1960) Window: $\lambda(u) = (1 - u) \cos \pi u + \frac{1}{\pi} \sin \pi u $					
$\frac{2\pi(1 + \cos \omega)}{(\pi^2 - \omega^2)^2}$	2	$\pi/2$	0.586	0.734	$1.23 \frac{1}{M}$
Generators Not of Truncated Form					
Daniell Window: $\lambda(u) = \frac{\sin \pi u}{\pi u}$					
$\frac{1}{2\pi}, \omega \leq \pi$	2	$\pi^2/6$	1	1.28	$\frac{1}{M}$
Bartlett-Priestley Window: $\lambda(u) = \frac{3}{(\pi u)^2} \left(\frac{\sin \pi u}{\pi u} - \cos \pi u \right)$					
$\frac{3}{4\pi} \left(1 - \left(\frac{\omega}{\pi} \right)^2 \right), \omega \leq \pi$	2	$\pi^2/10$	1.2	1.19	$\frac{2}{3} \frac{1}{M}$
Parzen-Cogburn-Davis Window: $\lambda(u) = \frac{1}{1 + u^{2r}}, s = \pi/2r$					
	$2r$	1	$\frac{s}{\sin s} \frac{2r-1}{r}$		
For $r = 2$:					
$\frac{1}{2} e^{- \omega /\sqrt{2}} \sin \left(\frac{ \omega }{\sqrt{2}} + \frac{\pi}{4} \right)$	4	1	1.66	1.66	$.45 \frac{1}{M}$

Figure 3.6. Eight Spectral Windows for $M = 20$.

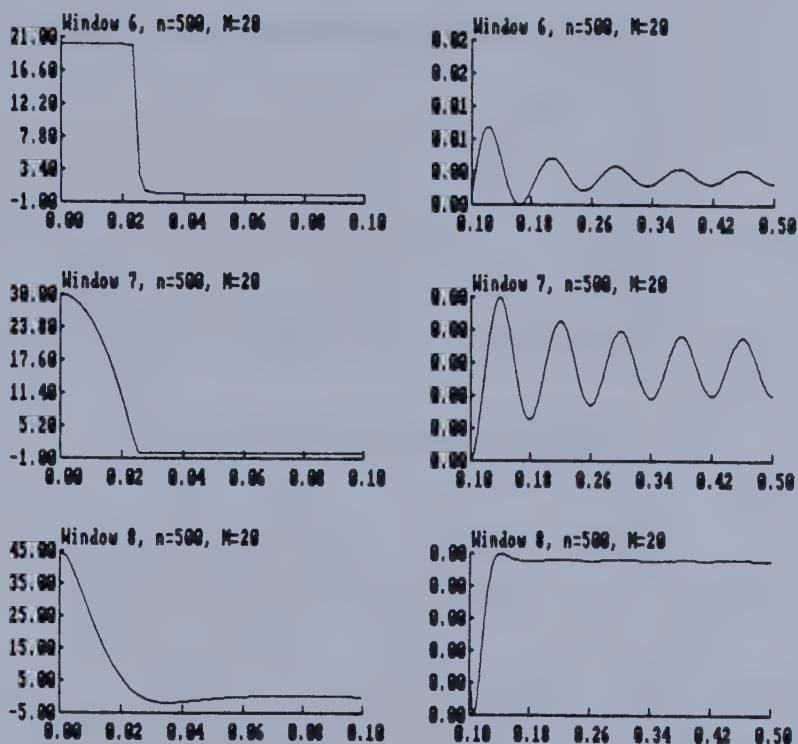


Figure 3.6 (continued)

the array **rho** which contains the sample autocorrelations to be used (M of them for the first five windows and $n-1$ of them for the last three). If **ioptw** is 6, 7, or 8, then the sample size **n** must also be input. The argument **c** is an output real variable and is used to find confidence intervals for the spectral density. We describe this further below Theorem 3.3.1.

3.3.3. Sampling Properties of Estimators

As we have pointed out, the properties of a spectral estimator based on smoothing \hat{f} will depend on the smoothness of f and the properties of the weighting function being used. In the case of a lag window of scale parameter form, the properties of the weighting function will depend on the scale parameter M and the lag window generator λ .

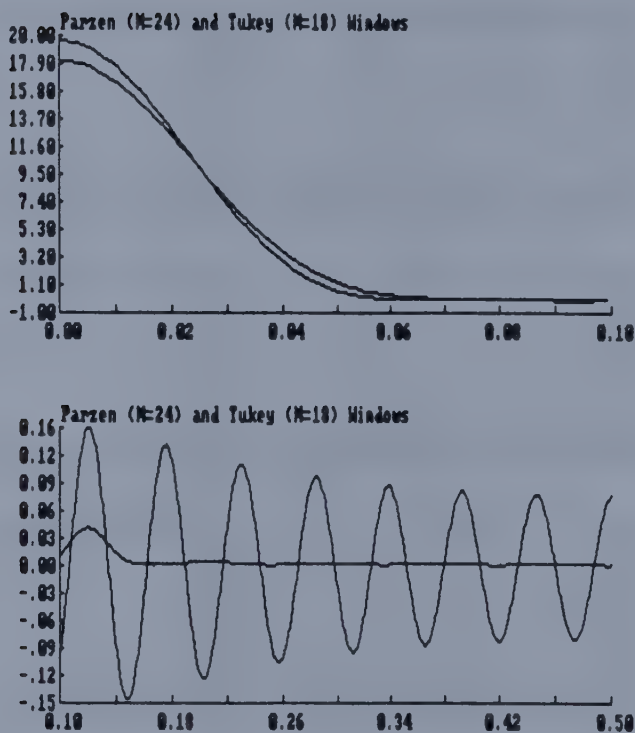


Figure 3.7. The Parzen and Tukey Spectral Windows for Comparable Bandwidths.

Measuring the Smoothness of f

In general, the smoothness of f can be measured by how its derivatives behave. For example, if f has a large peak at frequency ω , then $f''(\omega)$ will be very large in the region near frequency ω (see Problem C3.13). Thus we can index the smoothness of f by the number of finite derivatives that it has.

Definition. A spectral density f is said to be p -differentiable if

$$\sum_{v=-\infty}^{\infty} |v|^p |R(v)| < \infty.$$

The generalized p th derivative of f is given by

$$f^{[p]}(\omega) = \sum_{v=-\infty}^{\infty} |v|^p R(v) e^{-2\pi i v \omega}.$$

Note that if f is p -differentiable, then the p th derivative does in fact exist since, for example,

$$\begin{aligned} \sum_{v=-\infty}^{\infty} \left| \frac{d}{d\omega} R(v) \cos 2\pi v \omega \right| &= \sum_{v=-\infty}^{\infty} |R(v) 2\pi v \sin 2\pi v \omega| \\ &= 2\pi \sum_{v=-\infty}^{\infty} |v| |R(v)| |\sin 2\pi v \omega| \\ &\leq 2\pi \sum_{v=-\infty}^{\infty} |v| |R(v)| \\ &< \infty. \end{aligned}$$

Note also that the spectral density of an ARMA process is p -differentiable for all integers p (see Problem T3.8).

We should be able to adequately smooth \hat{f} using some truncation point and lag window generator λ as long as f doesn't have a peak so sharp relative to the width of λ that the smoothing won't work.

Measuring the Smoothness of λ

The character of λ can be measured by what is called its characteristic exponent and characteristic value.

Definition. The characteristic exponent q of the lag window generator λ is the largest integer k such that

$$\lim_{u \rightarrow 0} \frac{1 - \lambda(u)}{|u|^k}$$

exists, is finite, and is nonzero. If such a q exists, then the characteristic value of λ is given by

$$\lambda^{(q)} = \lim_{u \rightarrow 0} \frac{1 - \lambda(u)}{|u|^q}.$$

If the limit exists and is finite for all integers k , then we say that $q = \infty$.

Note that q measures how sharply λ can fall off as ω varies away from zero; that is, it measures the curvature of λ in the neighborhood of zero. For example, if $q = 2$, then $1 - \lambda(u)$ behaves like a quadratic function of u near zero. Note from Table 3.3 that the truncated periodogram window has $q = \infty$, the Bartlett window has $q = 1$, the next five windows have $q = 2$, and finally the Parzen-Cogburn-Davis window has $q = 2r$.

The next theorem summarizes a large number of basic results about the spectral estimators introduced in this section. The theorem is primarily concerned with estimators of scale parameter form. Part (g) allows us to make inferences about f for spectral windows of general form. Complete descriptions of these results can be found in Anderson (1971), Chapter 9 or Priestley (1981), Chapter 6. Many of the results of the theorem were found originally by Parzen (1957a, 1957b, 1958). After the statement of the theorem we describe how the results are used in practice.

Theorem 3.3.1

PROPERTIES OF WINDOW ESTIMATORS

Let $X(1), \dots, X(n)$ be a sample realization from a covariance stationary time series X which has spectral density function f which is p -differentiable for some $p \geq 1$, and let λ be a lag window generator having characteristic exponent $q > 0$ and characteristic value $\lambda^{(q)}$. Let

$$\hat{f}_{\lambda, M}(\omega) = \sum_{v=-\infty}^{\infty} \lambda\left(\frac{v}{M}\right) \hat{R}(v) e^{-2\pi i v \omega}$$

where M is chosen as a function of n so that $M \rightarrow \infty$ as $n \rightarrow \infty$. Note that the limits on the sum are actually $-M$ to M if λ is of truncated form and $-(n-1)$ to $n-1$ otherwise. Then

a) The estimator $\hat{f}_{\lambda, M}(\omega)$ is asymptotically unbiased, that is,

$$\lim_{n \rightarrow \infty} E \left[\hat{f}_{\lambda, M}(\omega) - f(\omega) \right] = 0.$$

b) ASYMPTOTIC BIAS. The rate of decay of bias in $\hat{f}_{\lambda, M}(\omega)$ is determined by:

i) If $p \geq q$ and M is chosen so that $M^q/n \rightarrow 0$ for $p \geq 1$ or $M^{q+1-p}/n \rightarrow 0$ for $p \leq 1$, then

$$\lim_{n \rightarrow \infty} M^q \left[E(\hat{f}_{\lambda, M}(\omega)) - f(\omega) \right] = -\lambda^{(q)} f^{[q]}(\omega).$$

ii) If $p < q$ and M is chosen so that $M^p/n \rightarrow 0$ for $p \geq 1$ or $M/n \rightarrow 0$ for $p \leq 1$, then

$$\lim_{n \rightarrow \infty} M^p \left[E(\hat{f}_{\lambda, M}(\omega)) - f(\omega) \right] = 0.$$

c) ASYMPTOTIC VARIANCE. If X is stationary to fourth order with absolutely summable autocovariances and fourth-order cumulant function, and M is chosen so that $M \rightarrow \infty$ and $M/n \rightarrow 0$ as $n \rightarrow \infty$, then the rate of decay of the variance of $\hat{f}_{\lambda, M}(\omega)$ is determined by

$$\lim_{n \rightarrow \infty} \frac{n}{M} \text{Var}(\hat{f}_{\lambda, M}(\omega)) = \begin{cases} 2f^2(\omega) \int_{-\infty}^{\infty} \lambda^2(u) du, & \omega = 0, .5 \\ f^2(\omega) \int_{-\infty}^{\infty} \lambda^2(u) du, & \omega \neq 0, .5 \end{cases}$$

$$\lim_{n \rightarrow \infty} \frac{n}{M} \text{Cov}(\hat{f}_{\lambda, M}(\omega_1), \hat{f}_{\lambda, M}(\omega_2)) = 0, \quad \omega_1 \neq \omega_2.$$

d) ASYMPTOTIC MEAN SQUARE ERROR. If $p \geq q$ as in part (b), and the assumptions in part (c) are valid, then for $\omega \neq 0, .5$, the rate of decay in the mean square error in $\hat{f}_{\lambda, M}(\omega)$ as an estimator of $f(\omega)$ is determined by

$$\begin{aligned} \text{MSE}(\hat{f}_{\lambda, M}(\omega)) &= E \left[\hat{f}_{\lambda, M}(\omega) - f_{\lambda, M} \right]^2 = \text{Var}(\hat{f}_{\lambda, M}(\omega)) + \left[E(\hat{f}_{\lambda, M}(\omega)) - f(\omega) \right]^2 \\ &\doteq \frac{M}{n} f^2(\omega) \int_{-\infty}^{\infty} \lambda^2(u) du + \frac{1}{M^{2q}} (\lambda^{(q)} f^{[q]}(\omega))^2, \end{aligned}$$

both terms of which will go to zero at the same rate if M is chosen as $M = cn^{1/2q+1}$ for some constant c , in which case

$$\lim_{n \rightarrow \infty} n^{2q/(2q+1)} \text{MSE}(\hat{f}_{\lambda, M}(\omega)) = cf^2(\omega) \int_{-\infty}^{\infty} \lambda^2(u) du + \frac{1}{c^{2q}} (\lambda^{(q)} f^{[q]}(\omega))^2$$

and

$$\lim_{n \rightarrow \infty} n^{q/(2q+1)} \left[E(\hat{f}_{\lambda, M}(\omega)) - f(\omega) \right] = -\frac{\lambda^{(q)} f^{[q]}(\omega)}{c}.$$

For fixed characteristic exponent q , the minimum value of the mean square error as a function of M for a lag window generator λ is proportional to

$$E_{\min} = (\lambda^{(q)})^{1/q} \int_0^1 \lambda^2(u) du.$$

e) If X is a linear process with absolutely summable coefficients and independent, identically distributed errors, if $M \rightarrow \infty$ and $M/n \rightarrow \infty$ as $n \rightarrow \infty$, and we let

$$\hat{\mathbf{f}}_r = (\hat{f}_{\lambda, M}(\omega_1), \dots, \hat{f}_{\lambda, M}(\omega_r))^T \quad \text{and} \quad \mathbf{f}_r = (f(\omega_1), \dots, f(\omega_r))^T$$

for r fixed frequencies $\omega_1, \dots, \omega_r$, then

$$\sqrt{\frac{n}{M}}(\hat{\mathbf{f}}_r - \mathbf{f}_r) \xrightarrow{\mathcal{L}} N_r(\mathbf{0}_r, \mathbf{V}),$$

where the elements of \mathbf{V} are given in part (c).

f) If the conditions of parts (b) and (e) hold, if $\lim_{n \rightarrow \infty} n/M^{2p+1} < \infty$ for $p < q$ or $n/M^{2q+1} \rightarrow 0$ as $n \rightarrow \infty$ for $q \leq p$, and if $f(\omega) > 0$, then

$$\sqrt{\frac{n}{M}}[\log \hat{f}_{\lambda, M}(\omega) - \log f(\omega)] = \sqrt{\frac{n}{M}} \log \frac{\hat{f}_{\lambda, M}(\omega)}{f(\omega)} \xrightarrow{\mathcal{L}} N\left(0, \int_{-\infty}^{\infty} \lambda^2(u) du\right).$$

g) For a general spectral estimator of the form

$$\hat{f}_{n, k}(\omega) = \sum_{v=-(n-1)}^{n-1} k_n(v) \hat{R}(v) e^{-2\pi i v \omega},$$

we have that the random variable $\nu \hat{f}_{n, k}(\omega)/f(\omega)$ is approximately χ_ν^2 , where the equivalent degrees of freedom ν are given by

$$\nu = \frac{2n}{\sum_{v=-(n-1)}^{n-1} k_n^2(v)}.$$

If k_n is of scale parameter form with lag window generator λ , then

$$\nu = \frac{2n}{M \int_{-\infty}^{\infty} \lambda^2(u) du}.$$

Implications: There are three kinds of facts in this very complicated theorem. First, parts (a) and (b) show that $\hat{f}_{\lambda, M}(\omega)$ is asymptotically unbiased, while part (c) shows that its variance goes to zero if $M/n \rightarrow 0$ as $n \rightarrow \infty$, which together give that $\hat{f}_{\lambda, M}(\omega)$ is a consistent estimator of $f(\omega)$. Part (c) also shows that estimators at fixed unequal frequencies are asymptotically uncorrelated. Second, part (d) and the other information in parts (b) and (c) allow us to make statements about how rapidly $\hat{f}_{\lambda, M}(\omega)$ converges to $f(\omega)$ and also to compare lag window generators λ and scale parameters M . Parts (b) and (c) show that the bias and variance behave inversely as a function of M , with the bias being proportional to M^{-q} in the usual case where $p \geq q$; that is, the spectral density is smoother than the window, and the variance is proportional to M . Thus a smaller M gives a less variable estimator, but one having more bias.

For fixed M and q , the bias is proportional to the characteristic value $\lambda^{(q)}$ of the window being used, while the variance is proportional to $\int_0^1 \lambda^2(u) du$. For the windows in Table 3.3 having $q = 2$, there is no window that has minimum values for both of these characteristics, and in this sense there is no optimal window. Since $\hat{f}_{\lambda,M}(\omega)$ is in fact a biased estimator, a true measure of its performance as an estimator of $f(\omega)$ is its average squared distance from $f(\omega)$, that is, its mean square error which is given in part (d). In this part we see that for all possible windows having characteristic value q , the bias and expected mean square error go to zero at the rates of $n^{-q/(2q+1)}$ and $n^{-2q/(2q+1)}$. For $q = 2$ this means choosing M to be proportional to $n^{1/5}$ (see the related issue of choosing `rbins` in the `DENSITY` command), and the rates of decay of bias and mean square error are $n^{-2/5}$ and $n^{-4/5}$. This last result is often expressed as the "four-fifths rule". These considerations argue for windows having a larger value of q . This rules out the Bartlett window as it has $q = 1$. The second part of part (d) gives us a means of comparing windows for a fixed value of q by comparing their values of E_{min} . In this sense the Bohman window is optimal. There are a wide variety of other methods for comparing windows (see Chapters 6 and 7 of Priestley (1981)) but we recommend using either the Parzen or Bohman windows as they are nonnegative, are of truncated form so that one need only calculate M correlations, and have good E_{min} values. The remaining problem is the choice of M (see below).

The third kind of facts in the theorem are those about the asymptotic distribution of $\hat{f}_{\lambda,M}(\omega)$ given in parts (e)–(g). These allow us to find confidence intervals for $f(\omega)$ at an individual frequency ω . Part (f) also shows that $\log \hat{f}_{\lambda,M}(\omega)$ is approximately $N(\log f(\omega), \frac{M}{n} \int_{-\infty}^{\infty} \lambda^2(u) du)$; that is, the variance of $\log \hat{f}_{\lambda,M}(\omega)$ is approximately constant over frequency. Again this motivates plotting not $\hat{f}_{\lambda,M}(\omega)$ but $\log \hat{f}_{\lambda,M}(\omega)$.

The confidence intervals based on parts (e), (f), and (g) are given by

$$\hat{f}_{\lambda,M}(\omega) \pm Z_{\alpha/2} \sqrt{\frac{M}{N} \hat{f}_{\lambda,M}^2(\omega) \int_{-\infty}^{\infty} \lambda^2(u) du},$$

$$\left(\frac{1}{c} \hat{f}_{\lambda,M}(\omega), c \hat{f}_{\lambda,M}(\omega) \right),$$

where $c = \exp(Z_{\alpha/2} \sqrt{\frac{M}{n} \int_{-\infty}^{\infty} \lambda^2(u) du})$, and

$$\left(\frac{\nu \hat{f}_{n,k}(\omega)}{\chi_{\alpha/2}^2}, \frac{\nu \hat{f}_{n,k}(\omega)}{\chi_{1-\alpha/2}^2} \right),$$

respectively. In the `WINDOW` command the number returned in the argument `c` (if it is included) is the value of c in the confidence interval formula above. Thus to display an estimator and the confidence intervals at each frequency one can use:

```

spec=WINDOW(rho,R0,M,Q,iptw,n,c)
spec1=spec/c
specu=spec*c
PLOTSP(spec,Q,R0,spec1,Q,R0,specu,Q,R0)

```

These intervals are for f at a single frequency. A large sample simultaneous confidence band on f for all frequencies can be obtained by (see Woodrooffe and Van Ness (1967)) multiplying both terms in the confidence interval based on part (f) by $\exp((2 \log M)^{1/2})$.

The Choice of M

The choice of M determines the amount of smoothing done, with too large (small) a value resulting in undersmoothing (oversmoothing). Wahba (1980) discusses objectively choosing the amount of smoothing to do in a similar setting (estimating the logarithm of the spectral density using essentially windows of the Parzen-Cogburn-Davis form), but in general the choice of M is very difficult unless some information about the function being estimated is known. Basically, if f has a narrow peak, we would like the bandwidth of the spectral window to be narrow so that leakage doesn't occur. This however leads to undersmoothing in other frequencies. Thus nonparametric spectral estimators have trouble "resolving" peaks without introducing spurious peaks in other ranges. We will see later that parametric spectral estimators can solve this problem in many cases. In any event, the prevailing view on the choice of M is to try more than one value and use the resulting plots to make general statements about f . For example, Parzen (1967) suggests using three values, M_1 , M_2 , and M_3 , where

$$.05 \leq \frac{M_1}{n} \leq .10, \quad .10 \leq \frac{M_2}{n} \leq .25, \quad .25 \leq \frac{M_3}{n} \leq .75.$$

In Figure 3.8 (see Example 3.10), we give spectral estimates using the Parzen window for three data sets: (1) the critical radio frequencies data, (2) the monthly sunspot data, and (3) the magnitude of a star data. Note that the estimator with the most variation in each case is the one with the highest truncation point. Notice how similar the results are for the critical radio frequencies data and the monthly sunspot numbers except that the sunspot numbers do not have the peak at period 12 months that the critical radio frequencies have. Thus the critical radio frequencies data are probably being influenced by annual weather variables such as temperature. The low frequency peak in these two series is in fact the so-called sunspot cycle.

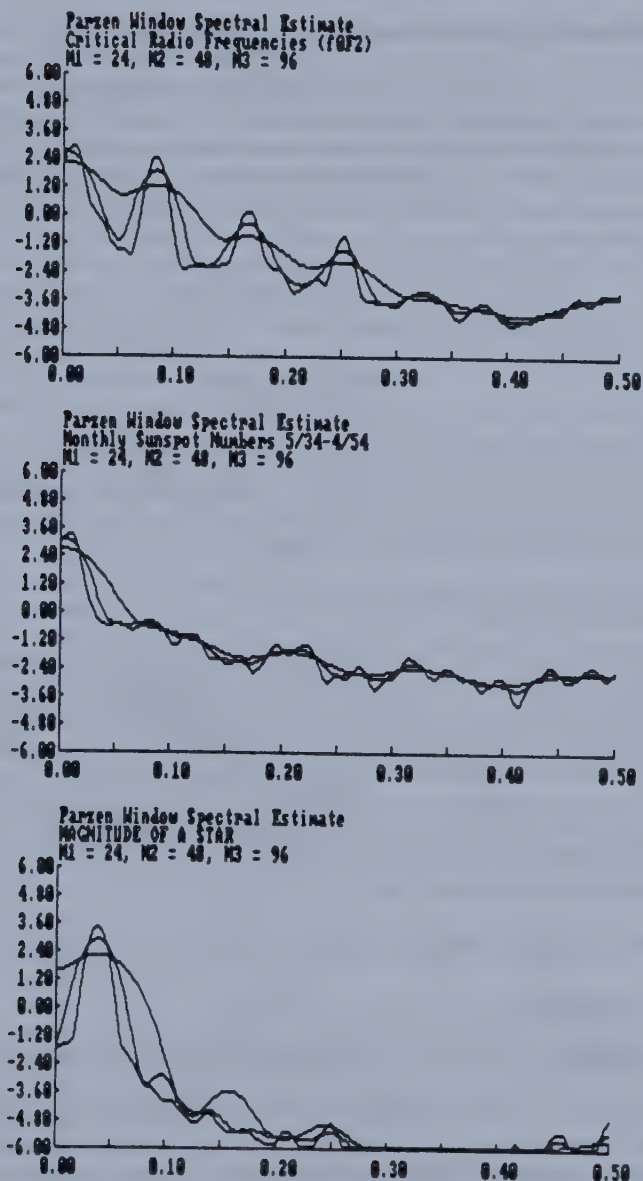


Figure 3.8. Parzen Window Estimates for Three Series, Each Using Three Different Truncation Points.

3.4. Estimating the Parameters of ARMA Models

In Section 2.5.5 we obtained the spectral density function of an ARMA process, while in Section 2.6.1 we showed how to forecast future values of the process if we knew the orders of the process and its parameters. Given data $X(1), \dots, X(n)$ from an ARMA process of orders (p, q) , we would like to find estimates $\hat{\alpha}$, $\hat{\beta}$, and $\hat{\sigma}^2$ of the parameters of the model as this would allow us to estimate f by

$$\hat{f}(\omega) = \hat{\sigma}^2 \frac{\left| \sum_{k=0}^q \hat{\beta}_k e^{2\pi i k \omega} \right|^2}{\left| \sum_{j=0}^p \hat{\alpha}_j e^{2\pi i j \omega} \right|^2},$$

and we could substitute the parameter estimates into the formulas for forecasting future values of X . In this section we assume that the orders p and q are known and discuss the estimation of the parameters. In the next section we consider the problem of determining p and q .

3.4.1. Maximum Likelihood Estimation

The first method that we consider is the application of the general principle of maximum likelihood. If we assume that the data are from a Gaussian ARMA process, we saw in Section 2.6 how the ARMA likelihood can be evaluated for specified values of its parameters via the Kalman Filter Algorithm. Much of the recent emphasis in calculating the maximum likelihood estimates for ARMA processes has been on combining a likelihood “evaluator” (see M  lard (1984)) with a nonlinear “optimizer” that requires no derivatives. This is in fact what the TIMESLAB command **DTARMA** does using the Nelder-Mead simplex algorithm as the optimizer (see O’Neill (1971)).

The DTARMA Command

The command

```
DTARMA(x,n,p,q,maxit,eps,alpha,beta,rvar,m2ll,ier,del)
```

will attempt to find the maximum likelihood estimators for an $\text{ARMA}(p, q)$ process given a realization of length n . The input integers **n**, **p**, and **q** contain n , p , and q , respectively, while the input arrays **x**, **alpha**, and **beta** contain the data and starting values for the coefficients. These starting values can be obtained from a variety of commands, including (1) method of moments estimates (see Section 3.4.3), (2) the **SEASEST** command (see Section 3.4.2), or (3) the **ARMASEL** command (see Section 3.5.3). In some cases, entering arrays of zeros as starting values will also lead to convergence of the likelihood maximization

procedure. The Nelder-Mead optimization procedure maximizes the likelihood with respect to $p + q$ variables, namely the elements of α and β . At each step, the algorithm calculates the likelihood at $p + q$ points, namely at the vertices of a simplex, that is, a $(p + q)$ -dimensional polygon. Then the algorithm tries to determine if there is a relative maximum of the function within this simplex. If so, then the simplex is made smaller and the procedure is done again. If not, then the simplex is moved and the procedure retried. At each step, the sample variance of the values of the likelihood function at the vertices of the simplex is calculated, and convergence is assumed if this variance is smaller than the convergence criterion that the user entered in the real scalar `eps`. The input integer `maxit` is the maximum number of evaluations of the likelihood function that the user is willing to allow; that is, `maxit` is $p + q$ times the number of steps of the algorithm that are to be allowed.

The quantities that are returned by `DTARMA` depend on what is returned in the integer `ier`. If `ier` = 0, then no errors were encountered, convergence was achieved, and the final values of the parameters of the process are returned in `alpha`, `beta`, and `rvar`. If `ier` = 1, then no errors were encountered, but convergence was not achieved. The values of the parameters when the algorithm stopped are returned. Thus `DTARMA` can be called again using these values as starting values. If `ier` is greater than one, then an error was encountered. If `DTARMA` succeeds in finding a maximum, then the resulting estimates for the AR and MA coefficients are guaranteed to have characteristic polynomials whose zeros are all outside the unit circle. This is because within the command, we have actually parametrized the model in terms of the partial autocorrelations corresponding to the coefficients of the AR part and the coefficients of the MA part, and then also transformed these in such a way that they were guaranteed to be between -1 and 1 (see Jones (1980)).

The basic theoretical results for maximum likelihood estimation for time series are contained in the following theorem, due originally to Whittle (1951).

Theorem 3.4.1 PROPERTIES OF MLE'S FOR TIME SERIES

Let $\mathbf{X}_n = (X(1), \dots, X(n))^T$ be a finite realization from a Gaussian time series that has spectral density f which depends on the parameters $\theta = (\theta_1, \dots, \theta_r)^T$. Let $\hat{\theta}_n$ be the maximum likelihood estimator of θ based on \mathbf{X}_n . Then under certain regularity conditions (see Dzhaparidze (1986)),

$$\sqrt{n}(\hat{\theta}_n - \theta) \xrightarrow{\mathcal{L}} N_r(0_r, \mathbf{I}^{-1}(\theta)),$$

where $\mathbf{I}(\theta)$ is called the information matrix of θ and has (j, k) th element

$$I_{jk}(\theta) = \frac{1}{2} \int_0^1 \frac{\partial \log f(\omega)}{\partial \theta_j} \frac{\partial \log f(\omega)}{\partial \theta_k} d\omega, \quad j, k = 1, \dots, r.$$

The regularity conditions are not easy to write down, but we note that there is no difficulty as long as we are considering cases where f is strictly positive. Thus for ARMA processes we will assume that X is invertible and therefore that f is positive.

Asymptotic Distribution of ARMA MLE's

The next theorem applies Theorem 3.4.1 to ARMA processes.

Theorem 3.4.2 ASYMPTOTIC DISTRIBUTION OF ARMA MLE's

Suppose X is an invertible Gaussian ARMA($p, q, \alpha, \beta, \sigma^2$) process. Let R_Y and R_Z be the autocovariance functions of an AR($p, \alpha, 1$) and AR($q, \beta, 1$) process, respectively. Let $\theta = (\alpha^T, \beta^T, \sigma^2)^T$. Then

a) The information matrix of θ is given by

$$I(\theta) = \begin{bmatrix} I^{\alpha\alpha} & I^{\alpha\beta} & 0_p \\ & I^{\beta\beta} & 0_q \\ \text{Symmetric} & & \frac{1}{2\sigma^4} \end{bmatrix}$$

where for $j, k = 1, \dots, p$ and $l, m = 1, \dots, q$, we have

$$I_{jk}^{\alpha\alpha} = R_Y(|j - k|)$$

$$I_{lm}^{\beta\beta} = R_Z(|l - m|)$$

$$I_{jl}^{\alpha\beta} = - \int_0^1 \frac{e^{2\pi i(j-l)\omega}}{g(e^{2\pi i\omega})h(e^{-2\pi i\omega})} d\omega.$$

b) If $q = 0$, then $I^{-1}(\alpha) = S(\alpha)$, where $S(\alpha)$ is the Schur matrix corresponding to α , and thus

$$\sqrt{n} \begin{pmatrix} \hat{\alpha} - \alpha \\ \hat{\sigma}^2 - \sigma^2 \end{pmatrix} \xrightarrow{\mathcal{L}} N_{p+1} \left(0_{p+1}, \begin{bmatrix} S(\alpha) & 0_p \\ 0_p^T & 2\sigma^4 \end{bmatrix} \right).$$

c) If $p = 0$, then $I^{-1}(\beta) = S(\beta)$, where $S(\beta)$ is the Schur matrix corresponding to β , and thus

$$\sqrt{n} \begin{pmatrix} \hat{\beta} - \beta \\ \hat{\sigma}^2 - \sigma^2 \end{pmatrix} \xrightarrow{\mathcal{L}} N_{q+1} \left(0_{q+1}, \begin{bmatrix} S(\beta) & 0_q \\ 0_q^T & 2\sigma^4 \end{bmatrix} \right).$$

Proof: If $X \sim \text{ARMA}(p, q, \alpha, \beta, \sigma^2)$, we have

$$\begin{aligned} \log(f(\omega)) &= \log \sigma^2 + \log(g(e^{2\pi i \omega})g(e^{-2\pi i \omega})) - \log(h(e^{2\pi i \omega})h(e^{-2\pi i \omega})) \\ &= \log \sigma^2 + \log g(e^{2\pi i \omega}) + \log g(e^{-2\pi i \omega}) - \log h(e^{2\pi i \omega}) - \log h(e^{-2\pi i \omega}), \end{aligned}$$

which gives

$$\frac{\partial \log f(\omega)}{\partial \sigma^2} = \frac{1}{\sigma^2}$$

$$\frac{\partial \log f(\omega)}{\partial \alpha_j} = \frac{1}{g(e^{2\pi i \omega})} e^{2\pi i j \omega} + \frac{1}{g(e^{-2\pi i \omega})} e^{-2\pi i j \omega} = \frac{1}{|g(e^{2\pi i \omega})|^2} \phi_j(\omega),$$

where

$$\phi_j(\omega) = e^{2\pi i j \omega} g(e^{-2\pi i \omega}) + e^{-2\pi i j \omega} g(e^{2\pi i \omega}).$$

Similarly,

$$\frac{\partial \log f(\omega)}{\partial \beta_j} = -\frac{1}{|h(e^{2\pi i \omega})|^2} \psi_j(\omega),$$

where

$$\psi_j(\omega) = e^{2\pi i j \omega} h(e^{-2\pi i \omega}) + e^{-2\pi i j \omega} h(e^{2\pi i \omega}).$$

From this we have, for example,

$$\begin{aligned} \frac{\partial \log f(\omega)}{\partial \alpha_j} \frac{\partial \log f(\omega)}{\partial \alpha_k} &= \frac{1}{|g(e^{2\pi i \omega})|^4} \phi_j(\omega) \phi_k(\omega) \\ &= \frac{e^{2\pi i(j+k)\omega}}{g^2(e^{2\pi i \omega})} + \frac{e^{-2\pi i(j+k)\omega}}{g^2(e^{-2\pi i \omega})} + \frac{e^{2\pi i(j-k)\omega}}{|g(e^{2\pi i \omega})|^2} + \frac{e^{-2\pi i(j-k)\omega}}{|g(e^{2\pi i \omega})|^2}. \end{aligned}$$

The integrals of the last two terms in this sum are $R_Y(|j-k|)$ by definition, while the integrals of the first two terms in the sum are zero since $1/g(z)$ can be expressed as a power series in only nonnegative powers of z (since the zeros of g are all outside the unit circle):

$$\frac{1}{g(z)} = \sum_{k=0}^{\infty} \gamma_k z^k$$

say, and thus, taking the first term for example,

$$\begin{aligned} \int_0^1 \frac{e^{2\pi i(j+k)\omega}}{g^2(e^{2\pi i \omega})} d\omega &= \int_0^1 e^{2\pi i(j+k)\omega} \sum_{l=0}^{\infty} \sum_{m=0}^{\infty} \gamma_l \gamma_m e^{2\pi i(l+m)\omega} d\omega \\ &= \sum_{l=0}^{\infty} \sum_{m=0}^{\infty} \gamma_l \gamma_m \int_0^1 e^{2\pi i(j+k+l+m)\omega} d\omega, \end{aligned}$$

and this integral is always zero since $j+k+l+m > 0$. Thus we have established the expression for $\mathbf{I}_{jk}^{\alpha\alpha}$. The other expressions in part (a) are established in the same way, while parts (b) and (c) follow easily from part (a).

The COEFFCSD Command

The COEFFCSD command finds the standard errors for the estimates of the coefficients of an ARMA process, that is, the square roots of the diagonal elements of $\mathbf{I}^{-1}(\theta)/n$. A 95% large sample confidence interval for θ_j is then given by adding and subtracting 1.96 times the standard error of θ_j from $\hat{\theta}_j$.

3.4.2. Approximate MLE's

Box and Jenkins (1970) show that if X is an invertible ARMA process with a Gaussian error term ϵ , then the likelihood function of α , β , and σ^2 for a realization \mathbf{X} of length n can be expressed as

$$L(\alpha, \beta, \sigma^2 | \mathbf{X}) = (2\pi\sigma^2)^{-n/2} |M_n(\alpha, \beta)|^{1/2} \exp\left(-\frac{S(\alpha, \beta)}{2\sigma^2}\right),$$

where M_n is a matrix that is a function of α and β , and

$$S(\alpha, \beta) = \sum_{t=-\infty}^n \hat{\epsilon}^2(t)$$

with $\hat{\epsilon}(t) = E(\epsilon(t) | \mathbf{X})$, and the conditional expectation is calculated assuming that α and β are the true values of the ARMA parameters. To find the α , β , and σ^2 that maximize the likelihood, Box and Jenkins state first that the term $|M_n(\alpha, \beta)|$ can be ignored, and second that the summation for $S(\alpha, \beta)$ can be truncated below at some finite limit, $-T$, say. We describe their suggestion for doing the truncation. Since

$$\epsilon(t) = \sum_{j=0}^p \alpha_j X(t-j) - \sum_{k=1}^q \beta_k \epsilon(t-k),$$

we can write the recursion

$$\hat{\epsilon}(t) = \sum_{j=0}^p \alpha_j E(X(t-j) | \mathbf{X}) - \sum_{k=1}^q \beta_k \hat{\epsilon}(t-k).$$

There are two (related) difficulties in calculating $\hat{\epsilon}(t)$ for given values of α and β . First, starting values for $\hat{\epsilon}$ must be obtained. Second, if $1 \leq t-j \leq n$, we have $E(X(t-j) | \mathbf{X}) = X(t-j)$, but for $t-j < 1$, we must determine $E(X(t-j) | \mathbf{X})$. Box and Jenkins suggest a procedure that solves both problems at the same

time. First they note that for an invertible ARMA model, $g(L)X(t) = h(L)\epsilon(t)$, $E(X(t)|\mathbf{X})$ is the same as $E(X(t)|\mathbf{X})$ for the “backward” model $g(L^{-1})X(t) = h(L^{-1})\eta(t)$ where η is also a $WN(\sigma^2)$ process with $\eta(t)$ independent of future X ’s, that is, $E(\eta(t)|X(t+j)) = 0$. This is based on what we saw in Section 2.4 about the finite memory forward and backward prediction coefficients.

Thus if we define $\hat{X}(t) = E(X(t)|\mathbf{X})$ and $\hat{\eta}(t) = E(\eta(t)|\mathbf{X})$, we have

$$\hat{X}(t) = \sum_{k=0}^q \beta_k \hat{\eta}(t+k) - \sum_{j=1}^p \alpha_j \hat{X}(t+j),$$

where $\hat{\eta}(t+j) = 0$ if $t+j < 1$ and $\hat{X}(t) = X(t)$ if $1 \leq t \leq n$. This gives the following scheme for evaluating $S(\alpha, \beta)$:

1. Let $\hat{\eta}(n-p+1) = \dots = \hat{\eta}(n-p+q) = 0$, and for $t = n-p, n-p-1, \dots, 1$, find

$$\hat{\eta}(t) = \sum_{j=0}^p \alpha_j X(t-j) - \sum_{k=1}^q \beta_k \hat{\eta}(t+k).$$

2. Calculate

$$\hat{X}(t) = \sum_{k=0}^q \beta_k \hat{\eta}(t+k) - \sum_{j=1}^p \alpha_j \hat{X}(t+j)$$

for $t = 0, -1, \dots, -T$, where $\hat{X}(t+j) = X(t+j)$ if $t+j \geq 1$, $\hat{\eta}(t+k) = 0$ if $t+k < 1$, and t is determined so that $\hat{X}(t)$ is essentially zero for $t < -T$.

3. Next take $\hat{X}(t) = \hat{\epsilon}(t) = 0$ for $t < -T$ and run the recursion forward; that is,

$$\hat{\epsilon}(t) = \sum_{j=0}^p \alpha_j \hat{X}(t-j) - \sum_{k=1}^q \beta_k \hat{\epsilon}(t-k), \quad t = -T, \dots, n.$$

Note that steps 2 and 3 are called back forecasting (or “backcasting”), and we are ensured that a truncation point $-T$ will be reached because we are running a difference equation (which has zeros inside the unit circle) backward and the forcing terms $\hat{\eta}$ become zero when their indices become less than one.

Once $\hat{\alpha}$ and $\hat{\beta}$ are found to minimize $S(\alpha, \beta)$, the value of σ^2 that maximizes the likelihood (again ignoring the term $|M_n|^{1/2}$) is given by

$$\hat{\sigma}^2 = \frac{S(\hat{\alpha}, \hat{\beta})}{n}.$$

The SEASEST Command

The command

SEASEST(w,n,ords,coeffs,lags,it,eps,nback,rvar,ier,sds,e)

will find the approximate MLE's for the parameters of a data set W of length n that is input in the array \mathbf{w} of length n where W follows a multiplicative subset ARMA model (see Section 2.5.6) of the form

$$\left(\sum_{j=0}^p \alpha_j L^j \right) \left(\sum_{k=0}^P \theta_k L^{u_k} \right) [W(t) - \mu] = \left(\sum_{l=0}^q \beta_l L^l \right) \left(\sum_{m=0}^Q \gamma_m L^{v_m} \right) \epsilon(t).$$

The input array **ords** contains the five elements p, P, q, Q, M , where M is 1 if μ is to be included in the model and 0 if not, while the array **lags** contains u_1, \dots, u_P followed by v_1, \dots, v_Q if either P or Q is nonzero (an argument must be included for **lags** whether such an array is defined or not). On input, the array **coeffs** contains initial estimates for the coefficients of the model in the order α, θ, β , and γ followed by one for the mean μ . Starting values need not be included for terms not in the model. On output, **coeffs** contains the final estimates of the parameters. If $M = 1$ in the array **ords**, then the final element of **coeffs** is the approximate MLE of the constant τ_0 in the rewritten form

$$g(L)G(L)W(t) = \tau_0 + h(L)H(L)\epsilon(t), \quad \tau_0 = \left(\sum_{j=0}^p \alpha_j \right) \left(\sum_{k=0}^P \theta_k \right) \mu$$

of the model.

The **SEASEST** command uses the Marquardt algorithm (see Box and Jenkins (1970), p. 504) which is an adaptation of a standard nonlinear regression algorithm using numerical derivatives. The input integer **it** is the number of iterations to allow in the procedure. Usually **it**=20 is sufficient. Note that if **it** is negative, then the estimates for each iteration are displayed. If **it**=1, then only one iteration is performed; that is, the output arguments **sds**, **rvar**, and **e** are formed while **coeffs** is unchanged. This allows the user to essentially evaluate the sum of squares function for several values in the neighborhood of coefficients judged to be the maximizing ones.

The input real scalar **eps** is a convergence criterion. If the largest difference of successive iterates is less than **eps**, the algorithm is judged to have converged. The input integer **nback** is how far back in the past that the user will allow the back forecasting to go. Thus **nback** is the number T in the description of the evaluation of S above. If **nback**=0, then no back forecasting is done. Usually **nback**=20 or 30 is sufficient.

In addition to **coeffs**, the arguments **rvar**, **ier**, **sds**, and optionally **e** are output from **SEASEST**. The real scalar **rvar** is the estimate of σ^2 . As is

usually the case with regression-type procedures, we do not use the divisor n in finding $rvar$, but rather n minus the number of estimated coefficients. The array sds contains estimates of the standard errors of the corresponding elements of the $coeffs$ array. Thus approximate 95% confidence intervals for the coefficients can be obtained by $coeffs-sds2$ and $coeffs+sds2$ where $sds2=1.96*sds$. The output integer ier is an error indicator. If $ier=0$, then convergence was reached, while $ier=1$ means that one of the matrices in the iterative procedure was judged to be singular. Values of ier of 3 or 4 mean that the diagonal adjustments in the system of equations in the algorithm become too large or too small respectively, but that there is a good chance that the outputted coefficients are satisfactory. If this happens, calling **SEASEST** with $coeffs$ slightly different than the output values can usually get it to terminate properly. The optional output argument e contains one step ahead prediction errors corresponding to $W(1), \dots, W(n)$.

Estimates for the Airline Data

The following macro produces estimates of the parameters of the model

$$W(t) = (1 + \beta L)(1 + \gamma L^{12})\epsilon(t)$$

for the airline data, where $W(t) = (1 - L)(1 - L^{12})\log(X(t))$.

```

1 ;;
2 ;;   AIREST.MAC: macro to find approximate MLE's for the
3 ;;               airline data.
4 ;;
5 PAUSE
6 ;start
7 READ(air,x,n)      ;read
8 w=LOGE(x,n)        ; and
9 w=DIFF(n,1,w)      ;   transform
10 nm1=n-1            ;
11 w=DIFF(nm1,12,w)   ;
12 nm13=n-13         ;
13 ;
14 ;   Initial Estimates:
15 ;
16 beta=<1,12,13>
17 ARMASEL(w,nm13,40,20,0,2,-2,.95,p,q,alpha,beta,rvar,ier)
18 ;
19 ;   MLE's:
20 ;
21 ords=<0,0,1,1,0>
22 lags=<12>
23 coeffs=<beta[1],beta[12]>
24 SEASEST(w,nm13,ords,coeffs,lags,-10,.001,30,rvar,ier,sds)
25 LIST(ier,rvar)
26 LIST(sds)

```

Note that the **ARMASEL** command (see Section 3.5.3) is used to find initial estimates of the coefficients. This command finds estimates of subset ARMA models. The model for W can be written as $W(t) = \sum_{l=0}^{13} \beta_l \epsilon(t-l)$, where $\beta_1 = \beta$ and $\beta_2 = \gamma$ and so we take the coefficients of lags 1 and 12 from the output of **ARMASEL** as the starting values in **SEASEST**. Actually, using zeros as the starting values also leads to the same output values. This is often the case with **SEASEST**. Note that no constant term is included in the model as the differencing will eliminate it. The output from the macro is given in Table 3.4 below.

Table 3.4. Estimates for the Airline Data

MA part of Select ARMA										
1	-.37	.00	.00	.00	.00	.00	.00	.00	.00	.00
11	.00	-.47	.14							
Iterations										
1	-.3697	-.4705								
2	-.3950	-.6184								
3	-.3904	-.6132								
Coefficients From SEASEST										
1	-.390353	-.613179								
ier=0										
rvar=.001363										
Standard Errors From SEASEST										
1	.083061	.073945								

3.4.3. Method of Moments Estimators

In Section 2.6 we described methods for finding the coefficients and error variance for AR, MA, and ARMA processes given their autocorrelations. These procedures are implemented in the **CORRAR**, **CORRMA**, and **CORRARMA** commands, respectively. A natural way to estimate the coefficients of these processes is to use estimates of the autocorrelations in these procedures. Such estimators are called method of moments estimators. For AR processes, the method of moments estimates use the Yule-Walker equations with sample autocorrelations substituted for true autocorrelations. The resulting estimates are thus called the Yule-Walker estimates and we will discuss them in detail in Section 3.4.4. For now, we just point out that for an $AR(p)$ process the Yule-Walker estimates have the same asymptotic properties as the maximum likelihood estimators; that is, they are asymptotically efficient. This is not true for the MA and ARMA case (see Priestley (1981), p. 355, for example).

MA Processes

As was pointed out in Section 2.6, the methods for MA and ARMA processes involve factoring an autocovariance generating function. When we use sample autocovariances in the procedures, we are not guaranteed that the factorization is feasible. For an MA process, the factorization is feasible if the estimate

$$\hat{f}(\omega) = \sum_{v=-q}^q \hat{R}(v) \cos 2\pi v\omega$$

is positive for all $\omega \in [0, 1]$.

3.4.4. Estimation for AR Processes

Autoregressive processes are widely used in both spectral estimation (see Section 3.8) and forecasting. In this section we consider some non-MLE estimation procedures for such processes. Throughout the section we assume that the sample mean has been removed from the data being analyzed.

Yule-Walker Estimators

Perhaps the most natural method of estimating the parameters α and σ^2 of an $\text{AR}(p, \alpha, \sigma^2)$ process is to substitute the sample autocovariances \hat{R} for the true autocovariances R in the Yule-Walker equations (see part (c) of Theorem 2.5.3), and then solve the resulting sample Yule-Walker equations:

$$\hat{\Gamma}_p \hat{\alpha} = -\hat{r} \quad \text{and} \quad \hat{\sigma}^2 = \sum_{j=0}^p \hat{\alpha}_j \hat{R}(j),$$

to obtain what are called the Yule-Walker estimators (YWE) $\hat{\alpha}$ and $\hat{\sigma}^2$. Note that this method is known as the “autocorrelation method” in the engineering literature (see Makhoul (1975)). The availability of Levinson’s algorithm makes the YWE’s attractive from a computational point of view. Unfortunately there is wide evidence that the YWE’s perform poorly in some cases (see Example 3.12, Tjøstheim and Paulsen (1983), and Newton and Pagano (1983b) for more information about this problem). We will see that they are ordinary least squares estimators for the regression model

$$X(t) = -\alpha_1 X(t-1) - \cdots - \alpha_p X(t-p), \quad t = 1, \dots, n+p,$$

where the unobserved X ’s (the ones having indices $1-p, \dots, 0$ and $n+1, \dots, n+p$) are replaced by their expectation which is zero. Thus the YWE’s multiply the infinitely long realization $\{X(t), t \in \mathcal{Z}\}$ by the “boxcar” function that is one for $1 \leq t \leq n$ and zero elsewhere. We will see that this leads to some very simple calculation procedures. However, it is clear that if zeros are poor proxies for the unobserved X ’s, the resulting estimators can be poor.

To derive the calculation procedures we introduce the circular shift operator.

Definition. Let \mathbf{x} be an m -dimensional vector. Then the vector circular shift operator L applied to \mathbf{x} results in the m -dimensional vector \mathbf{z} given by

$$\mathbf{z} = L\mathbf{x} = \begin{pmatrix} x(n) \\ x(1) \\ \vdots \\ x(n-1) \end{pmatrix}.$$

The operator applied to a matrix \mathbf{W} results in a matrix \mathbf{U} whose columns are the result of applying L to each of the columns of \mathbf{W} .

Theorem 3.4.3

PROPERTIES OF YWE'S

Let $\hat{\alpha}$ and $\hat{\sigma}^2$ be the YWE's calculated from a sample $X(1), \dots, X(n)$. Then

a) $\hat{\alpha}$ and $\hat{\sigma}^2$ are the ordinary least squares estimators for the regression problem $\mathbf{y} = -\mathbf{X}\alpha + \epsilon$ where $E(\epsilon) = \mathbf{0}_n$, $\text{Var}(\epsilon) = \sigma^2 \mathbf{I}_n$, \mathbf{y} is the $(n+p)$ -dimensional vector $\mathbf{y} = (X(1), \dots, X(n), 0, \dots, 0)^T$, and \mathbf{X} is the $(n+p) \times p$ matrix $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_p)$, where $\mathbf{x}_j = L^j \mathbf{y}$.

b) The zeros of $\hat{g}(z) = \sum_{j=0}^p \hat{\alpha}_j z^j$ are guaranteed to be outside the unit circle.

c) If $X(1), \dots, X(n)$ is a sample from a Gaussian $\text{AR}(p, \alpha, \sigma^2)$ process, then

$$\sqrt{n} \begin{pmatrix} \hat{\alpha} - \alpha \\ \hat{\sigma}^2 - \sigma^2 \end{pmatrix} \xrightarrow{\mathcal{L}} N_{p+1} \left(\mathbf{0}_{p+1}, \begin{bmatrix} \mathbf{S}(\alpha) & \mathbf{0}_p \\ \mathbf{0}_p^T & \frac{1}{2\sigma^4} \end{bmatrix} \right),$$

where $\mathbf{S}(\alpha)$ is the Schur matrix corresponding to α .

Proof: The inclusion of the p zeros at the end of \mathbf{y} makes it so that

$$(\mathbf{X}^T \mathbf{X})_{jk} = (L^j \mathbf{y})^T (L^k \mathbf{y}) = n \hat{R}(|j-k|)$$

$$(\mathbf{X}^T \mathbf{y})_j = (L^j \mathbf{y})^T \mathbf{y} = n \hat{R}(j),$$

and thus the normal equations $\mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\alpha}} = -\mathbf{X}^T \mathbf{y}$ become $\hat{\Gamma}_p \hat{\boldsymbol{\alpha}} = -\hat{\mathbf{r}}_p$. Further, the usual sum of squares of residuals becomes

$$\text{RSS} = \mathbf{y}^T \mathbf{y} + \hat{\boldsymbol{\alpha}}^T \mathbf{X}^T \mathbf{y} = n\hat{R}(0) + n \sum_{j=1}^p \hat{\alpha}_j \hat{R}(j) = n\hat{\sigma}^2,$$

that is, $\hat{\sigma}^2 = \text{RSS}/n$.

Part (b) follows from the positive definiteness of \hat{R} and from part (b) of Theorem 2.6.6.

Implications: This theorem gives two important results in addition to verifying our observation about the use of zeros as proxies for unobserved data. First, the fitted process is guaranteed to be stable. Thus the spectral density is sure to be positive. Further, if we use the $\hat{\boldsymbol{\alpha}}$ in the AR prediction formula, we have

$$\hat{X}(n+j) = - \sum_{j=1}^p \hat{\alpha}_j \hat{X}(n+j-1), \quad j = 1, 2, \dots,$$

and the predictors will converge to zero, rather than diverging or oscillating between $-\infty$ and ∞ (see Theorem 1.6.1). The second result is that the YWE's are asymptotically efficient; that is, they have the same large sample properties as the maximum likelihood estimators.

Before turning to alternatives to the YWE's, we discuss the simple algorithms that exist for finding the YWE's. We do this because their derivation helps to motivate the alternatives. We first present a theorem (due to Cybenko (1983)) from linear algebra that will make the derivation simple. By the notation $\mathbf{y}|\mathbf{X} \Rightarrow (\hat{\boldsymbol{\beta}}, \mathbf{e}, S)$, we mean that the results of the regression of the vector \mathbf{y} on the matrix \mathbf{X} are the least squares coefficients $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$, residuals $\mathbf{e} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$, and sum of squares of residuals $S = \mathbf{e}^T \mathbf{e}$.

Theorem 3.4.4 CYBENKO'S THEOREM

Suppose \mathbf{X} is an $(n \times k)$ matrix of full rank k such that

$$\mathbf{y}|\mathbf{X} \Rightarrow (\hat{\boldsymbol{\alpha}}, \mathbf{e}, S) \quad \text{and} \quad \mathbf{z}|\mathbf{X} \Rightarrow (\hat{\boldsymbol{\beta}}, \mathbf{f}, T)$$

and the matrix $(\mathbf{y}, \mathbf{X}, \mathbf{z})$ is of full rank $k+2$. Then

$$\text{a) } \mathbf{y}|(\mathbf{X}, \mathbf{z}) \Rightarrow \left(\begin{bmatrix} \hat{\boldsymbol{\alpha}} - b\hat{\boldsymbol{\beta}} \\ b \end{bmatrix}, \mathbf{e} - b\mathbf{f}, S - b^2T \right)$$

$$b) \mathbf{z} | (\mathbf{y}, \mathbf{X}) \Rightarrow \left(\begin{bmatrix} c \\ \hat{\beta} - c\hat{\alpha} \end{bmatrix}, \mathbf{f} - c\mathbf{e}, T - c^2 S \right),$$

where

$$b = \frac{\mathbf{f}^T \mathbf{e}}{\mathbf{f}^T \mathbf{f}} \quad \text{and} \quad c = \frac{\mathbf{f}^T \mathbf{e}}{\mathbf{e}^T \mathbf{e}}.$$

This theorem is important because of the special form of \mathbf{X} in the regression model solved by the YWE's. Let $\hat{\alpha}_1, \dots, \hat{\alpha}_m$ and $\hat{\sigma}_1^2, \dots, \hat{\sigma}_M^2$ be the YWE's of orders $1, \dots, m$ given data $X(1), \dots, X(n)$. Then it is clear that $\hat{\alpha}_j$ and $\hat{\sigma}_j^2$ are the solution of the regression $\mathbf{y}_m | -\mathbf{X}_j$, where \mathbf{y}_m is the $(n+m)$ -dimensional vector $\mathbf{y}_m = (X(1), \dots, X(n), 0, \dots, 0)^T$ and \mathbf{X}_j consists of the first j columns of the $(n+m) \times m$ matrix $\mathbf{X}_m = (L\mathbf{y}_m, \dots, L^m \mathbf{y}_m)$. Thus the regression problem for order $j+1$ is the same as that for order j except that one more column (namely $L^{j+1} \mathbf{y}_m$) is added.

Let $\hat{\theta}_1, \dots, \hat{\theta}_m$ represent the negative of the last elements of $\hat{\alpha}_j, \dots, \hat{\alpha}_m$. Then in analogy with the partial autocorrelation of a time series, we call $\hat{\theta}_j$ the sample partial autocorrelation coefficient of lag j . The following theorem gives a very simple algorithm for calculating the $\hat{\theta}_j$'s. The algorithm is essentially Levinson's algorithm except that it accepts as input the data \mathbf{y} rather than the autocovariances \hat{R} . It provides a method for doing the regression calculations by triangularizing \mathbf{X} rather than $\mathbf{X}^T \mathbf{X}$ (see Stewart (1973) for a description of the benefits in general of operating on \mathbf{X} rather than $\mathbf{X}^T \mathbf{X}$). This algorithm is referred to as a lattice filter algorithm in the engineering literature.

Theorem 3.4.5 A "GIVEN DATA" LEVINSON ALGORITHM

Let $\mathbf{y}_m = (\mathbf{y}^T, 0_m^T)^T$ where $\mathbf{y}^T = (X(1), \dots, X(n))$, and let

$$\mathbf{x}_j = L^j \mathbf{y}_m, \quad j = 1, \dots, m.$$

Let $\mathbf{e}_0 = \mathbf{y}_m$, $\mathbf{f}_0 = \mathbf{x}_1$, and $S_0 = n\hat{R}(0) = \mathbf{y}_m^T \mathbf{y}_m$. Let

$$\mathbf{X}_p = -(\mathbf{x}_1, \dots, \mathbf{x}_p), \quad p = 1, \dots, m,$$

and

$$\mathbf{y}_m | \mathbf{X}_p \Rightarrow (\hat{\alpha}_p, \mathbf{e}_p, S_p), \quad p = 1, \dots, m$$

$$\mathbf{x}_{p+1} | \mathbf{X}_p \Rightarrow (\hat{\beta}_p, \mathbf{f}_p, T_p), \quad p = 1, \dots, m-1.$$

Then

a) $\hat{\beta}_p = \mathbf{P}_p \hat{\alpha}_p, \quad p = 1, \dots, m-1.$

b) *The coefficients satisfy the recursion*

$$\hat{\alpha}_p = \begin{cases} a_1, & p = 1 \\ \begin{bmatrix} \hat{\alpha}_{p-1} + a_p \mathbf{P}_p \hat{\alpha}_{p-1} \\ a_p \end{bmatrix}, & p = 2, \dots, m, \end{cases}$$

where

$$a_p = -\frac{\mathbf{f}_{p-1}^T \mathbf{e}_{p-1}}{\mathbf{f}_{p-1}^T \mathbf{f}_{p-1}}, \quad p = 1, \dots, m.$$

c) *The sums of squares of residuals satisfy*

$$S_p = T_p = S_{p-1}(1 - a_p^2), \quad p = 1, \dots, m.$$

d) *The forward and backward residuals satisfy*

$$\mathbf{e}_p = \mathbf{e}_{p-1} + a_p \mathbf{f}_{p-1}, \quad p = 1, \dots, m$$

$$\mathbf{f}_p = L(\mathbf{f}_{p-1} + a_p \mathbf{e}_{p-1}), \quad p = 1, \dots, m-1.$$

Proof: The theorem is clearly true for $p = 1$. For $p = 2, \dots, m$, we seek the results of the regression $\mathbf{y}_m | (-\mathbf{x}_1, \dots, -\mathbf{x}_p)$. Making the identification $\mathbf{y} = \mathbf{y}_m$, $\mathbf{z} = -\mathbf{x}_p$, and $\mathbf{X} = (-\mathbf{x}_1, \dots, -\mathbf{x}_{p-1})$, and noting that

$$-\mathbf{x}_p | \mathbf{X}_{p-1} \Rightarrow (-\hat{\beta}_{p-1}, -\mathbf{f}_{p-1}, T_{p-1}),$$

we have from the first part of Cybenko's Theorem that

$$\hat{\alpha}_p = \begin{pmatrix} \hat{\alpha}_{p-1} - b(-\hat{\beta}_{p-1}) \\ b \end{pmatrix},$$

where $b = -\mathbf{f}_{p-1}^T \mathbf{e}_{p-1} / \mathbf{f}_{p-1}^T \mathbf{f}_{p-1} = a_p$, that is,

$$\hat{\alpha}_p = \begin{pmatrix} \hat{\alpha}_{p-1} + a_p \hat{\beta}_{p-1} \\ a_p \end{pmatrix}.$$

Thus to show part (b) we need only show part (a). For $p = 2, \dots, m-1$, we have

$$\begin{aligned} \hat{\beta}_p &= -(\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T \mathbf{x}_{p+1} = (\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{P}_p \mathbf{X}_p^T \mathbf{y}_m \\ &= \mathbf{P}_p (\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{P}_p \mathbf{P}_p \mathbf{X}_p^T \mathbf{y}_m \\ &= \mathbf{P}_p (\mathbf{X}_p^T \mathbf{X}_p)^{-1} \mathbf{X}_p^T \mathbf{y}_m = \mathbf{P}_p \hat{\alpha}_p, \end{aligned}$$

since $\mathbf{P}_p^2 = \mathbf{I}_p$, $\mathbf{P}_p \mathbf{A}^{-1} \mathbf{P}_p = \mathbf{A}^{-1}$ if \mathbf{A} is Toeplitz, and

$$\mathbf{X}_p^T \mathbf{x}_{p+1} = -n(\hat{R}(p), \dots, \hat{R}(1))^T = -n\mathbf{P}_p(\hat{R}(1), \dots, \hat{R}(p))^T = \mathbf{P}_p \mathbf{X}_p^T \mathbf{y}_m.$$

To show part (c), note that $\mathbf{y}_m^T \mathbf{y}_m = \mathbf{x}_p^T \mathbf{x}_p$ and $\mathbf{X}_{p-1}^T \mathbf{y}_m = \mathbf{P}_{p-1} \mathbf{X}_{p-1}^T \mathbf{x}_p$ and so

$$\begin{aligned} S_{p-1} &= \mathbf{y}^T \mathbf{y} - \hat{\alpha}_{p-1}^T \mathbf{X}_{p-1}^T \mathbf{y}_m \\ &= \mathbf{x}_p^T \mathbf{x}_p - \hat{\beta}_{p-1}^T \mathbf{P}_{p-1} \mathbf{P}_{p-1} \mathbf{X}_{p-1}^T \mathbf{x}_p \\ &= \mathbf{x}_p^T \mathbf{x}_p - \hat{\beta}_{p-1}^T \mathbf{X}_{p-1}^T \mathbf{x}_p \\ &= T_{p-1}, \end{aligned}$$

which means again by Cybenko's Theorem that

$$S_p = S_{p-1} - a_p^2 T_{p-1} = S_{p-1}(1 - a_p^2) = T_p.$$

Part (a) of Cybenko's Theorem gives the expression for \mathbf{e}_p in part (d). To verify the expression for \mathbf{f}_p we use its second part. Define the matrix $\mathbf{W}_p = (-\mathbf{y}_m, \mathbf{X}_{p-1})$ and note that $L(\mathbf{W}_p) = \mathbf{X}_p$. Thus

$$\begin{aligned} \mathbf{f}_p &= \mathbf{x}_{p+1} - \mathbf{X}_p \hat{\beta}_p \\ &= L(\mathbf{x}_p) - L(\mathbf{W}_p) \hat{\beta}_p \\ &= L(\mathbf{x}_p - \mathbf{W}_p \hat{\beta}_p). \end{aligned}$$

Now the coefficients of the regression $\mathbf{x}_{p+1} | \mathbf{X}_p$ (which are $\hat{\beta}_p$) are also the coefficients for the regression $\mathbf{x}_p | \mathbf{W}_p$ since $\mathbf{W}_p^T \mathbf{W}_p = n\hat{\Gamma}_p = \mathbf{X}_p^T \mathbf{X}_p$, and $\mathbf{W}_p^T \mathbf{x}_p = n(\hat{R}(p), \dots, \hat{R}(1))^T = \mathbf{X}_p^T \mathbf{x}_{p+1}$. Thus $\mathbf{x}_p - \mathbf{W}_p \hat{\beta}_p$ are the residuals of regressing \mathbf{x}_p on \mathbf{W}_p , and making the identification $\mathbf{z} = \mathbf{x}_p$, $\mathbf{y} = -\mathbf{y}_m$, and $\mathbf{X} = \mathbf{X}_{p-1}$ in the second part of Cybenko's Theorem gives $\mathbf{x}_p - \mathbf{W}_p \hat{\beta}_p = \mathbf{f}_{p-1} + a_p \mathbf{e}_{p-1}$.

The PARCORR Command

Theorem 3.4.5 shows that the sample partial autocorrelation coefficient $\hat{\theta}_{p+1}$ is the sample correlation coefficient for the arrays \mathbf{e}_p and \mathbf{f}_p since the mean of each of these arrays is zero (they are residuals from regression problems where the observation vectors \mathbf{y}_m and $\mathbf{x}_{p+1} = L^{p+1} \mathbf{y}_m$ have mean zero since we have assumed that the mean has been subtracted from $X(1), \dots, X(n)$) and $\mathbf{e}_p^T \mathbf{e}_p = \mathbf{f}_p^T \mathbf{f}_p$. But the j th elements of the arrays \mathbf{e}_p and \mathbf{f}_p are the residuals of regressing $X(j)$ on $-X(j-1), \dots, -X(j-p)$ and $X(j-p-1)$ on $-X(j-p), \dots, -X(j-1)$, respectively. Thus \mathbf{e}_p and \mathbf{f}_p are one step ahead, order p , forward and backward

sample prediction errors, respectively. This is in keeping with the definition of the theoretical partial autocorrelation of lag p .

Recalling that $\hat{\theta}_j = -\hat{\alpha}_j(j)$ and $\hat{\sigma}_j^2 = S_j/n$, we have $\mathbf{e}_0 = \mathbf{y}_m$, $\mathbf{f}_0 = L(\mathbf{y}_m)$, and $\hat{\sigma}_0^2 = \hat{R}(0)$, and for $j = 1, \dots, m$:

$$\hat{\theta}_j = \frac{\mathbf{f}_{j-1}^T \mathbf{e}_{j-1}}{\mathbf{f}_{j-1}^T \mathbf{f}_{j-1}}$$

$$\hat{\sigma}_j^2 = \hat{\sigma}_{j-1}^2 (1 - \hat{\theta}_j^2)$$

$$\mathbf{e}_j = \mathbf{e}_{j-1} - \hat{\theta}_j \mathbf{f}_{j-1}$$

$$\mathbf{f}_j = L(\mathbf{f}_{j-1} - \hat{\theta}_j \mathbf{e}_{j-1}).$$

Given the partials, one can easily obtain the estimated coefficients for an AR(p) model by using the **PARTAR** command.

Properties of Sample Partial Autocorrelations

The sample partial autocorrelations play an important role in determining models for time series. In the next theorem we describe some basic sampling properties that they have.

Theorem 3.4.6

PROPERTIES OF PARTIAL AUTOCORRELATIONS

If X is a Gaussian AR(p) process, then the sample partial autocorrelations for lags $p + 1$ and higher are asymptotically independent and identically normally distributed, are asymptotically unbiased, and have asymptotic variance $1/n$.

This theorem suggests that a natural method for choosing the order of an autoregressive process is that order for which all of the partials of higher lag are outside the interval $\pm 2/\sqrt{n}$. Several authors (see Box and Jenkins (1970), p. 178, for example) state that the sample partial autocorrelations for a general time series have sampling properties similar to those of the ordinary sample autocorrelations.

The Burg and Related Algorithms

We have now seen that the estimator $\hat{\theta}_{p+1}$ can be interpreted as a sample correlation coefficient, a sample partial autocorrelation coefficient, or the regression coefficient for regressing \mathbf{e}_p on \mathbf{f}_p . Thus a whole family of possible AR coefficient estimators can be obtained by constructing alternative correlation

coefficients or regression coefficients from \mathbf{e}_p and \mathbf{f}_p , and then using Levinson's algorithm to obtain the corresponding coefficients. The best known example of this is a method due to Burg (1967) (see also Ulrych and Bishop (1975)). It differs from the YWE's in two ways. First, the inner products are not calculated over the whole length of \mathbf{e}_p and \mathbf{f}_p , but rather only over their $(p+1)$ st through n th elements. This appears to avoid the problem of assigning zeros to the unobserved data values. Second, the Burg procedure uses a different method of estimating a correlation coefficient, namely

$$\tilde{\theta}_{p+1} = -\frac{\tilde{\mathbf{f}}_p^T \tilde{\mathbf{e}}_p}{\frac{1}{2}[\tilde{\mathbf{f}}_p^T \tilde{\mathbf{f}}_p + \tilde{\mathbf{e}}_p^T \tilde{\mathbf{e}}_p]},$$

where $\tilde{\mathbf{f}}_p$ and $\tilde{\mathbf{e}}_p$ denote vectors consisting of only the $(p+1)$ st through n th elements of \mathbf{f}_p and \mathbf{e}_p . Note that if we remove the tildes from $\tilde{\mathbf{f}}_p$ and $\tilde{\mathbf{e}}_p$ we would get $\hat{\theta}_{p+1}$ back again since $\mathbf{f}_p^T \mathbf{f}_p = \mathbf{e}_p^T \mathbf{e}_p$.

The formula for $\tilde{\theta}_{p+1}$ can be motivated from both a correlation and regression point of view. First, $\tilde{\theta}_{p+1}$ is the value of a that minimizes

$$S(a) = \|\tilde{\mathbf{f}}_p - a\tilde{\mathbf{e}}_p\|^2 + \|\tilde{\mathbf{e}}_p - a\tilde{\mathbf{f}}_p\|^2$$

(see Problem T3.13). Second, from a correlation point of view, Stuart (1955) has shown that for a random sample from a bivariate normal distribution with zero means and equal but unknown variances,

$$\hat{\rho} = \frac{\sum_{i=1}^n x_i y_i}{\frac{1}{2} \left[\sum_{i=1}^n x_i^2 + \sum_{i=1}^n y_i^2 \right]}$$

is an asymptotically efficient estimator of the correlation coefficient of X and Y . Note that the true forward and backward prediction variances are indeed equal while the prediction errors have mean zero.

There are other algorithms that follow the strategy of obtaining a partial autocorrelation coefficient and then using Levinson's algorithm to determine the corresponding AR coefficients (see Makhoul (1977)). Note that as long as the partials used are less than one in absolute value, then the resulting polynomial $\hat{g}(z)$ is guaranteed to have all of its zeros outside the unit circle.

Least Squares Estimates for AR Coefficients

In the previous section we saw how the Yule-Walker estimators for an $AR(p)$ are the least squares estimators for the regression of $X(t)$ on the previous p X 's for $t = 1, \dots, n + p$, with unobserved X 's replaced by zeros. Another natural way to estimate the parameters is to do the regression only for $t = p + 1, \dots, n$, thus eliminating the need for replacing any unobserved variables. This leads to what are often referred to as least squares estimators of the coefficients and error variance. In the engineering literature this method is known as the "autocovariance method." It is often stated that there is no Levinson-type recursive algorithm for finding least squares estimators for successively higher orders. However such a recursion has been developed in the signal processing literature (see DeMeersman (1975)), and Ensor (1987) has discussed the algorithm from a statistical point of view. However, the least squares estimators are still rarely used because they are not guaranteed to lead to positive spectra or stable predictors.

The following macro will find least squares estimators for an $AR(p)$.

```

1 ;;
2 ;;  LSAR.MAC: macro to find least squares estimates for
3 ;;          an AR(p) process given data x(1),...,x(n).
4 ;;
5 ;;  INPUT: p,n,x
6 ;;
7 ;;  OUTPUT: alpha,rvar (rvar is RSS/n)
8 ;;
9 PAUSE
10 ;start
11 nmp=n-p
12 nmpm1=nmp-1
13 pp1=p+1
14 y=EXTRACT(x,pp1,n) ;observation vector is x(p+1),...,x(n)
15 xx=<0>              ;we will accumulate design matrix in xx
16 i=1                ;we'll do it backward
17 ;                  ;when i is 1 we'll do last column
18 ;startloop
19 ;
20 nl=nmpm1+i
21 x1=EXTRACT(x,i,nl)
22 xx=<x1,xx>          ;put this column before the others
23 IF(i.eq.p,endoloop) ;finished?
24 i=i+1              ;no
25 GOTO(startloop)
26 ;
27 ;endloop           ;yes
28 ;
29 xx=-xx
30 REG(y,xx,nmp,p,alpha,rss,t,res) ;do the regression
31 rvar=rss/n

```

```
32 LIST(alpha)
```

```
33 LIST(rvar)
```

3.4.5. Other ARMA Estimators

There are a wide variety of other estimation procedures that have been proposed for ARMA processes. Many of these consist of expressing the model as a regression model wherein $X(t)$ is regressed on the previous p X 's and ϵ 's at the time points $t - 1, \dots, t - q$. Since we don't know the ϵ 's, they are approximated by the residuals of a high order autoregressive model fit to the X 's. The best known example of this idea is due to Durbin (1959, 1960). The stepwise ARMA procedure described in Section 3.5 below also uses this idea.

Other ARMA estimation procedures include those proposed by Hannan (1970), Parzen (1971), Anderson (1975), and Tsay and Tiao (1984).

3.5. Identifying ARMA Models

The problem of determining what type of ARMA model best fits a data set has become very important in time series analysis. In this section we describe a variety of diagnostics that have proven useful in determining models and orders of models. We will then describe a TIMESLAB macro that will help guide a user through parts of the model selection problem.

3.5.1. Some Useful Diagnostics

Suppose that we have a realization $X(1), \dots, X(n)$ from a covariance stationary time series X . We saw in Chapter 2 that the true autocorrelations, partial autocorrelations, and spectral density have certain characteristics for different model types. The statements below are of necessity rather general and somewhat vague. In later parts of this section we will describe some methods that have been developed to try to use these statements in a somewhat automatic way.

AR Processes

The autocorrelation function decays to zero and then oscillates about zero, while the partial autocorrelation becomes identically zero for lags greater than the true order p . The decay of ρ can follow a sinusoidal pattern if some of the zeros of its characteristic polynomial are complex. The spectral density of an AR process can contain very sharp peaks, while the troughs appear somewhat less sharp.

MA Processes

The autocorrelation function is identically zero for lags greater than the order of the process, while the partial autocorrelation function does not become identically zero. The peaks in the spectral density are smooth relative to what can be attained for AR models, while the troughs can be rather sharp.

ARMA Process

Here the autocorrelation function decays in the same way as that of an AR process but only after lag q . Again the partial autocorrelation does not become zero. The spectral density function can now have either sharp or rounded peaks and troughs.

The statements made above are for models that have nonzero coefficients. If only a few of the coefficients of a model are nonzero, then further statements can be made. We consider two examples of this, although there are a wide variety of possibilities. First, if we have an MA or ARMA model with only a few nonzero coefficients, then $\rho(v)$ can be small for some lags smaller than q (see the sales data example in Section 3.6). Second, if X is an AR(p) with only α_p nonzero, then the spectral density of X will have a very sharp peak at frequency $1/p$ (see Problem C3.20). We will consider below automatic methods for determining if a subset model will adequately represent a time series.

Other Diagnostics

In addition to the sample autocorrelations $\hat{\rho}$, sample partial autocorrelations $\hat{\theta}$, and sample spectral density \hat{f} , we will consider several other quantities.

The Biased and Unbiased Residual Variances

The “biased residual variances” $\hat{\sigma}_j^2$ and “unbiased residual variances” $\tilde{\sigma}_j^2$ are defined by $\hat{\sigma}_0^2 = \hat{R}(0)$ and

$$\hat{\sigma}_j^2 = \hat{\sigma}_{j-1}^2(1 - \hat{\theta}_j^2) = \frac{\text{RSS}(j)}{n}$$

$$\tilde{\sigma}_j^2 = \frac{n}{n-j} \hat{\sigma}_j^2,$$

where $\text{RSS}(j)$ is the sum of squares of residuals for the j th-order Yule-Walker regression model. These quantities are calculated for $j = 1, \dots, M$ for some integer M which is chosen to be larger than any lag for which there is useful information in the sample correlation function. One rule of thumb that we have found useful for selecting M is as the smaller of $n/4$ and 30.

The biased residual variances are so called because they correspond to using the divisor n in the regression model instead of the usual divisor $n - j$

that is used to produce an unbiased estimate of error variance in regression models. The unbiased residual variance is in fact the residual sum of squares divided by $n - j$. In fact, in small samples, neither of these estimators is unbiased, while asymptotically they are both unbiased. Note that since the $\hat{\theta}_j$'s are between -1 and 1 , the biased residual variances are a monotonically nonincreasing sequence, while the unbiased residual variances can in fact attain a relative minimum since they are the product of the decreasing sequence $\hat{\sigma}_j^2$ and the increasing sequence $n/(n - j)$.

It is sometimes useful to standardize the biased and unbiased residual variances by dividing them by $\hat{R}(0)$. This converts the biased residual variances to numbers between 0 and 1 and essentially produces quantities that are analogous to $1 - R^2$ in regression.

Quantities for the Dual Process

If X has a spectral density that is strictly positive, then we saw in Section 2.5.8 that its reciprocal f_i exists and is the spectral density of some process Y . We will call Y the process that is dual to X . Given $X(1), \dots, X(n)$, we can produce quantities for the dual process analogous to those we have defined so far by estimating the inverse autocovariance function $\hat{R}i$, that is, the autocovariance function of Y , by

$$\hat{R}i(v) = \frac{1}{Q} \sum_{j=1}^Q \frac{1}{\hat{f}(\omega_j)} \cos 2\pi v \omega_j,$$

for $v = 0, \dots, M$, and then fitting autoregressive models of successively higher orders and finding what are called the inverse partials and the inverse biased residual variances and inverse unbiased residual variances. If it is felt that X is either an AR process or an invertible MA or ARMA process, then we can also estimate $\hat{R}i(v)$ by

$$\tilde{R}i(v) = \frac{1}{\hat{\sigma}_K^2} \sum_{j=0}^{K-v} \hat{\alpha}_K(j) \hat{\alpha}_K(j+v).$$

In fact, one indication that X might not be an invertible process is that $\hat{R}i$ and $\tilde{R}i$ do not agree very well. We saw in Section 2.5.8 that if X is an AR(p) or MA(q) process, then the inverse of its spectral density is the spectral density of an MA(p) or AR(q) process, respectively. If X is an ARMA(p, q) process, then Y is an ARMA(q, p) process.

The Generalized Partial Autocorrelations

We saw in Theorem 2.6.4 that the true partial autocorrelation of lag v is the negative of the v th prediction coefficient for the one step ahead memory- v linear predictor. Since these prediction coefficients satisfy the Yule-Walker equations (with the signs reversed), we can view θ_v as the negative of the last coefficient of an $AR(v)$ model fit to X . For an $AR(p)$ process, this last coefficient is zero for $v > p$. Woodward and Gray (1981) have extended this idea to ARMA processes. They define the generalized partial autocorrelation coefficient (GPAC), $\theta_{k,j}$, to be the negative of the k th autoregressive coefficient when an $ARMA(k,j)$ model is fit to X . They show that if X is in fact an $ARMA(p,q)$ process, then

$$\theta_{k,j} = \begin{cases} \theta_k, & k = p, j \geq q \\ 0, & k > p, j = q. \end{cases}$$

Thus if one were to create a table of $\theta_{k,j}$ for AR orders on the left between 0 and P and MA orders across the top between 0 and Q , then the true orders p and q could be identified by finding the place in the table where henceforth the values across the row are constant and the values down the column are all zero. The following macro will form the GPAC array for a user-specified correlation sequence (either population or sample values) and maximum AR and MA orders. Note that the GPAC values for $q = 0$ are the usual partial autocorrelations, while the values for $p = 0$ are not included. In the macro we have estimated the AR coefficients for each of the models involved by using the CORRARMA with estimated correlations. This leads to consistent estimates. One could also use the iterated autoregressions method of Tsay and Tiao (1984).

```

1 ;;
2 ;;   GPAC.MAC: macro to find the GPAC array for ARMA(p,q)
3 ;;   processes for p=1,...,pmax and q=0,...,qmax.
4 ;;   The macro will also display the resulting table
5 ;;   correctly as long as qmax is less than 7. The
6 ;;   AR orders (1 through pmax) will be on the left
7 ;;   and the MA orders (0 through qmax) across the top.
8 ;;
9 ;;   INPUT: pmax, qmax, rho(1),...,rho(pmax + qmax)
10 ;;
11 ;;
12 PAUSE
13 ;start
14 PROMPTOFF
15 ptq=pmax*(qmax+1)
16 gpac=LINE(ptq,0,0)           ;reserve space for the array
17 alpha1=CORRAR(rho,r0,pmax,rvar) ;find values for q=0
18 part2=ARPART(alpha1,pmax,ier)
19 gpac[1]=part2[1]

```

```

20 p=1
21 q=1
22 jj=1
23 ;
24 ;s1
25 ;
26 LIST(p,q)
27 rvar=corrarma(rho,r0,p,q,100,.001,ier,alpha1,beta1)
28 jj=jj+1
29 gpac[jj]=-alpha1[p]
30 IF(q.eq.qmax,s2)
31 q=q+1
32 GOTO(s1)
33 ;
34 ;s2
35 ;
36 IF(p.eq.pmax,s3)
37 p=p+1
38 jj=jj+1
39 gpac[jj]=part2[p]
40 q=1
41 GOTO(s1)
42 ;
43 ;s3
44 ;
45 qmaxp1=qmax+1
46 LABEL(gpac)='GPAC Array'
47 LIST(gpac,ptq,qmaxp1,8f7.3)
48 PROMPTON

```

We illustrate the use of the GPAC in Table 3.5 wherein we give the GPAC table (1) for the ARMA(3,2) process (see Woodward and Gray (1981))

$$X(t) - 1.5X(t-1) + 1.21X(t-2) - .455X(t-3) = \epsilon(t) + .2\epsilon(t-1) + .9\epsilon(t-2),$$

(2) for the sample autocorrelation function of a realization of length 200 from this process, and (3) for the log (base 10) of the lynx data. Note how the expected pattern occurs in the true GPAC and is discernible in the one for the simulated data. For the lynx data, it would appear that an AR(2) would be appropriate. In the next section we will consider the lynx data further.

The Prediction Variance Horizon Function

A number of the series that arise (particularly in business and economics) are well modeled as an ARMA process where only a few of the moving average coefficients are nonzero. A diagnostic that has proven useful in detecting such a model is what is called the prediction variance horizon function, defined as follows (see Parzen (1981)). We saw in Theorem 2.4.1 that the error variance

Table 3.5. The GPAC Arrays for the True and Sample Autocorrelations for an ARMA(3,2) Process and for the Log of the Lynx Data

GPAC Array for True ARMA(3,2) Model						
p\q	0	1	2	3	4	5
1	.845	.606	.391	.328	1.356	1.632
2	-.706	-.458	-.070	2.073	-.119	5.367
3	.414	.836	.455	.455	.455	.455
4	.299	.683	.000	.254	.156	.418
5	-.304	-.434	.000	.000	.420	.264
6	-.145	-.646	.000	.000	.000	.217
7	.245	.279	.000	.000	.000	.000
8	.062	.848	.000	.000	.000	.000
GPAC Array for Sample (n=200) from ARMA(3,2) Model						
p\q	0	1	2	3	4	5
1	.780	.467	.158	.704	4.741	1.377
2	-.625	-.358	.131	-.824	2.780	-1.489
3	.351	.878	.578	.578	.573	.490
4	.338	.491	.000	70.991	1.979	.062
5	-.167	-.605	.045	.044	6.816	-.651
6	-.223	-.305	-.140	-.154	-.167	-.249
7	.115	.225	-.078	.463	.024	-1.598
8	.057	.235	-.224	-.191	-.248	-.240
GPAC Array for Log (base 10) of lynx data						
p\q	0	1	2	3	4	5
1	.785	.433	-.389	3.733	1.256	.786
2	-.720	-.795	-.845	-.872	-.890	-.973
3	-.143	.872	-.955	.130	2.813	.300
4	-.206	-.283	-.586	.353	.386	.258
5	.115	.265	.536	-.056	-.186	-4.525
6	.085	-.196	.528	-.019	.896	-2.246
7	.208	.162	.146	-.296	-.309	-.465
8	.118	-.060	1.227	.067	1.245	-.164

of the optimal infinite memory, h step ahead forecast is given by

$$\sigma_h^2 = \sigma^2 \sum_{k=0}^{h-1} \beta_k^2,$$

where the β 's and σ^2 are the coefficients and error variance of the MA(∞) representation of X . We can estimate these quantities by first fitting a (possibly high order) autoregressive model to the data and then inverting the characteristic polynomial of the fitted process. If the data come from a process having a subset MA part, then we would expect that the estimates of the β 's would have "flat spots" corresponding to the zero MA coefficients.

We define the prediction variance horizon (PVH) as

$$\text{PVH}(h) = \frac{\sigma^2 \sum_{k=0}^{h-1} \beta_k^2}{R(0)}, \quad h = 1, 2, \dots$$

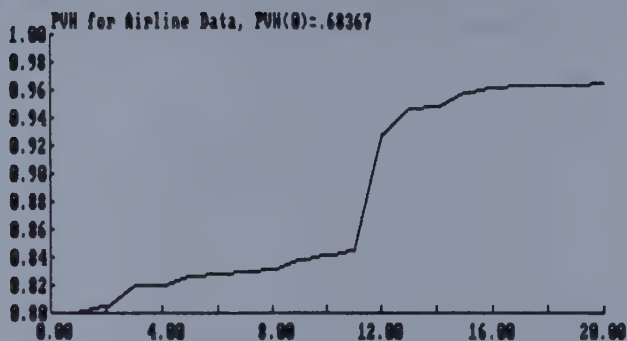


Figure 3.9. Values of PVH for the First and Twelfth Difference of the Log of the Airline Data.

Note that $0 \leq \text{PVH}(h) \leq 1$ since we also know that

$$\lim_{h \rightarrow \infty} \sigma_h^2 = R(0).$$

To illustrate the use of PVH, consider Figure 3.9 which contains the graph of the PVH for horizons 1 through 40 for the first and 12th differences of the log of the airline data (see Example 3.13 for the macro that produced the values of PVH). Note the jumps in the PVH for $h=1$ and 12. This suggests that the data contain a subset MA part having at least lags 1 and 12.

3.5.2. The AIC and Related Criteria

The problem of order determination in time series analysis is similar to the general statistical problem of determining which of a set of possible models best fits a set of data. One very important method for making such a choice is due to Akaike (1971) and is thus called Akaike's Information Criterion (AIC). Although there are a variety of ways to describe AIC, one simple motivation is as follows. If we have two competing models for a data set and they have the same number of parameters, then we would probably choose the one that best "fits" the data in some sense. For example, in regression, if we have two possible models for the dependent variable and each is in terms of a subset of q independent variables, then we would probably choose that model that leads to the smaller sum of squares of residuals. If competing models have different numbers of parameters, then we would not want to choose one based merely on goodness of fit. Again, in regression, it is a well-known phenomenon that adding a variable to a model is guaranteed to decrease the sum of squares of residuals, but also tends to increase the variability in predictions resulting from the larger model. We have also seen this phenomenon in the problems of smoothing that

we have considered in this book (see the moving average smoother in Chapter 1 and the problem of smoothing the periodogram in Section 3.2). In such problems, increasing the complexity of a model may reduce variability, while at the same time increasing bias.

Using general principles of information theory, Akaike proposed measuring the “goodness” of a model that has r parameters $\theta = (\theta_1, \dots, \theta_r)^T$ by

$$\text{AIC}(r) = -2 \log L(\hat{\theta}) + 2r,$$

where L is the likelihood function of θ , and $\hat{\theta}$ is the maximum likelihood estimate of θ . The first term is the measure of how well a model fits the data, while the second is a penalty for the number of parameters in the model. The AIC is then used to choose the model that has the smallest value of the criterion. Thus if a term in AIC is the same for all competing models, it can be deleted. For ARMA modeling, this leads to considering

$$\text{AIC}(r) = n \log \hat{\sigma}_r^2 + 2r,$$

where $\hat{\sigma}_r^2$ is the maximum likelihood estimate of the error variance for the model having r parameters.

Using AIC for AR Models

AIC is very easily applied to the problem of AR order determination. For example, the following macro will use it to choose the order of an AR model for a user-specified data set.

```

1 ;;
2 ;;   AIC.MAC: This macro finds aic(k) for k=1,...,m
3 ;;           for data x(1),...,x(n), and returns the
4 ;;           chosen order p.
5 ;;
6 ;;   INPUT: x,n,m
7 ;;
8 ;;   OUTPUT: p, aic0, aic(1),...,aic(m)
9 ;;
10 PAUSE
11 ;start
12 y=SUBMNS(x,n,1,xbar)      ;subtract mean
13 part=PARCORR(y,n,m,rvar) ;find sigma hats (rvar)
14 aic1=LOGE(rvar,m)
15 aic1=n*aic1
16 aic=LINE(m,0,2)
17 aic=aic+aic1
18 LABEL(aic)='AIC Criterion'
19 MAXMIN(aic,m,amax,imax,amin,p) ;find minimum value
20 rho=CORR(x,n,0,0,1,r0,per)      ;find aic(0)=n log(r0)
21 aic0=LOGE(r0)

```

```

22 aic0=n*aic0
23 IF(amin.lt.aic0,end)      ;check if it's really order 0
24 p=0
25 ;
26 ;end
27 ;
28 CLEAR(y,part,aic1,amax,imax,amin,r0,xbar,rvar)

```

For an AR process, the value of AIC for order 0 is just $AIC(0) = n \log \hat{R}(0)$. If we apply this macro to the critical radio frequencies data, we obtain the values of AIC in Table 3.6. Note that order 17 is chosen, with the value for order 14 being just a little larger than that for order 17. For the log of the lynx data, AIC chooses order 11 (see Problem C3.14).

Table 3.6. Values of AIC for the Critical Radio Frequencies Data

AIC Criterion					
1	45.178100	-17.545470	-16.150540	-15.371550	-14.322820
6	-27.734450	-30.535160	-38.921380	-47.714840	-72.871150
11	-80.831830	-91.284290	-124.621300	-134.903300	-132.925800
16	-133.348200	-135.238500	-133.812100	-131.959900	-130.779100
21	-128.786800	-127.054800	-126.048800	-124.377200	-125.346500
26	-123.501000	-121.502900	-119.560900	-117.599600	-115.663100
aic0=427.409300					

A large amount of effort has been made in investigating the properties of using the AIC for AR processes.

Theorem 3.5.1 INCONSISTENCY OF AIC FOR AN $AR(p)$

If X is a Gaussian $AR(p)$ process and \hat{p} is the order chosen by the AIC based on a realization of length n , where the maximum order allowed is K which is known to be greater than p , then

$$\lim_{n \rightarrow \infty} \Pr(\hat{p} = k) = \begin{cases} g(k, K) > 0, & k \geq p \\ 0, & k < p, \end{cases}$$

where $g(k, K)$ is a function only of k and K .

This result (due to Shibata (1976)) says that the AIC is not guaranteed to choose the right order no matter how long a realization one has. Thus it is said to be inconsistent. Shibata (1976) calculated the function $g(k, K)$ for $0 \leq p \leq k \leq K = 10$, and found that for $p = 0$ through 8 the asymptotic probability of correctly choosing the order was between 70 and 78% (see Problem T3.14).

Note further that the theorem also guarantees that AIC will not asymptotically underestimate the order. This will be important in autoregressive spectral estimation (see Section 3.8).

The Hannan-Quinn Consistent Modification of AIC

The AIC can be easily modified to produce a consistent estimate of the AR order. This fact is given in the following theorem due to Hannan and Quinn (1979).

Theorem 3.5.2 CONSISTENT AR ORDER DETERMINATION

If the $2k$ in the AIC is replaced by $2kc \log \log n$ for any constant $c > 1$, then the resulting order determining criterion will be consistent; that is, the probability of correctly choosing the order of the AR process converges to 1 as $n \rightarrow \infty$.

This theorem says that if we replace the penalty function $2k$ by one that is just a little larger (note that $\log \log n$ goes to ∞ very slowly, see Problem C3.15), then we no longer will tend to overestimate the order. Unfortunately, there is evidence that seems to indicate (see Hannan and Quinn (1979)) that in small and moderate sized samples, the consistent methods tend to underestimate the order.

The CAT Criterion and AR Models

Another criterion that is useful for determining the order of an autoregressive process is the CAT criterion due to Parzen (1977). Instead of assuming that X is an $\text{AR}(p)$ process, Parzen considered the problem of determining the order p of an autoregressive process that approximates in some optimal way the behavior of an arbitrary covariance stationary time series. We will discuss this further in Section 3.8.

Definition. Let $\hat{\sigma}_1^2, \dots, \hat{\sigma}_M^2$ be the error variance estimates from the Yule-Walker equations based on a sample of size n , and let

$$\tilde{\sigma}_j^2 = \frac{n}{n-j} \hat{\sigma}_j^2, \quad j = 1, \dots, M.$$

Then the criterion autoregressive transfer function (CAT) criterion for order k

is defined to be

$$\text{CAT}(k) = \begin{cases} \left(\frac{1}{n} \sum_{j=1}^k \frac{1}{\hat{\sigma}_j^2} \right) - \frac{1}{\hat{\sigma}_k^2}, & k = 1, \dots, M \\ - \left(1 + \frac{1}{n} \right) \hat{R}(0), & k = 0. \end{cases}$$

Note that the natural way to define $\text{CAT}(0)$ would be as just $-\hat{R}(0)$, in which case it has been shown that the AIC and the CAT criterion are asymptotically equivalent (see Bhansali (1986)). Parzen has suggested the definition above so that CAT can be used as a test for white noise. As we pointed out above, if X is an $\text{AR}(0)$ (that is, white noise) process, then AIC (and hence CAT with $\text{CAT}(0) = -\hat{R}(0)$) would asymptotically only choose order zero approximately 75% of the time. In Example 3.14, we consider the level of significance of the modified CAT criterion as a test for white noise.

AIC and ARMA Models

The AIC is also applicable for choosing the order of ARMA processes. Ideally, we should use the exact maximum likelihood method for each possible order and then calculate AIC based on the maximized likelihood. Unfortunately, this is very time consuming and in some cases (particularly when the process is close to nonstationary or noninvertible) it is difficult to get the maximization procedure to converge. An alternative is to use the estimate of the error variance obtained from the method of moments estimator for each order. This also can be difficult to implement since in some cases the factorization of the covariance generating function becomes infeasible (see Section 2.6.3). Thus in TIMESLAB, we suggest using the consistent but inefficient method described in Section 3.5.3 to estimate the error variance for each order (see the **ARMAAIC** macro described in Example 3.15). Hannan and Rissanen (1982) have considered the properties of this method of determining p and q . Once an order is determined, one can use the exact (**DTARMA**) or approximate (**SEASEST**) likelihood methods for estimating the parameters of the process.

3.5.3. The Stepwise ARMA Method

Since an ARMA model looks very much like an ordinary regression model, it is natural to apply regression techniques in their analysis. In fact we saw in Section 3.4.4 that applying ordinary least squares to autoregressive data leads to asymptotically efficient parameter estimates. In the MA case this was not true. In this section we extend the regression analogy to the ARMA case by describing the TIMESLAB command **ARMASEL** which performs a stepwise ARMA procedure. See Section A.5.4 for a general discussion of stepwise regression.

Suppose we feel that an ARMA model of some order (p, q) should fit a data set $X(1), \dots, X(n)$ and (P, Q) is an upper bound on the order. Then in analogy with stepwise regression analysis, we would write

$$X(t) = -\alpha_1 X(t-1) - \dots - \alpha_P X(t-P) + \beta_1 \epsilon(t-1) + \dots + \beta_Q \epsilon(t-Q) + \epsilon(t),$$

that is,

$$\mathbf{y} = \mathbf{W}\boldsymbol{\theta} + \boldsymbol{\epsilon},$$

where, for $r = \max(P, Q)$, we have

$$\mathbf{y} = (X(r+1), \dots, X(n))^T, \quad \boldsymbol{\theta} = (-\boldsymbol{\alpha}^T, \boldsymbol{\beta}^T)^T, \quad \boldsymbol{\epsilon} = (\epsilon(r+1), \dots, \epsilon(n))^T,$$

and $\mathbf{W} = (\mathbf{X}, \mathbf{E})$, where

$$\mathbf{X} = \begin{bmatrix} X(r) & \dots & X(r-P) \\ \vdots & & \vdots \\ X(n-1) & \dots & X(n-P-1) \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \epsilon(r) & \dots & \epsilon(r-Q) \\ \vdots & & \vdots \\ \epsilon(n-1) & \dots & \epsilon(n-Q-1) \end{bmatrix}.$$

If we knew the elements of \mathbf{E} , then we could form

$$\mathbf{A} = \begin{bmatrix} \mathbf{W}^T \mathbf{W} & \mathbf{W}^T \mathbf{y} \\ \mathbf{y}^T \mathbf{W} & \mathbf{y}^T \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T \mathbf{X} & \mathbf{X}^T \mathbf{E} & \mathbf{X}^T \mathbf{y} \\ \mathbf{E}^T \mathbf{X} & \mathbf{E}^T \mathbf{E} & \mathbf{E}^T \mathbf{y} \\ \mathbf{y}^T \mathbf{X} & \mathbf{y}^T \mathbf{E} & \mathbf{y}^T \mathbf{y} \end{bmatrix},$$

and then use ordinary stepwise regression to explore the problem of determining which elements of $\alpha_1, \dots, \alpha_P$ and β_1, \dots, β_Q are nonzero. There have been a number of suggestions for finding proxies for the ϵ 's, most of them equivalent to estimating the ϵ 's by one step ahead prediction errors for a high order AR process fit to the data. The **ARMASEL** command proceeds as follows. The matrix \mathbf{A} consists of lagged inner products of X 's and/or ϵ 's. Thus for large n , \mathbf{A} should be close to

$$\mathbf{A}_1 = \frac{1}{n} \begin{bmatrix} \Gamma_{XX} & \Gamma_{X\epsilon} & \mathbf{r}_{XX} \\ & \Gamma_{\epsilon\epsilon} & \mathbf{r}_{X\epsilon} \\ \text{Symmetric} & & R_{XX}(0) \end{bmatrix},$$

where

$$(\Gamma_{XX})_{j,k} = R_{XX}(|j-k|), \quad j, k = 1, \dots, P$$

$$(\Gamma_{X\epsilon})_{j,k} = R_{X\epsilon}(|j-k|), \quad j = 1, \dots, P, \quad k = 1, \dots, Q$$

$$(\Gamma_{\epsilon\epsilon})_{j,k} = R_{\epsilon\epsilon}(|j-k|), \quad j, k = 1, \dots, Q$$

$$(\mathbf{r}_{XX})_j = R_{XX}(j), \quad j = 1, \dots, P$$

$$(\mathbf{r}_{X\epsilon})_j = R_{X\epsilon}(-j), \quad j = 1, \dots, Q.$$

We will estimate these auto- and cross-covariances and then apply stepwise regression to the matrix $\hat{\mathbf{A}}_1$ which is the matrix \mathbf{A}_1 with the estimates replacing the true covariances.

If $X \sim \text{ARMA}(p, q, \alpha, \beta, \sigma^2)$ is invertible, we can write X as an infinite order AR process and thus for some possibly large order s we should have

$$\epsilon(t) = \sum_{j=0}^s \gamma_j X(t-j).$$

This gives by the Filter Theorem

$$R_{\epsilon\epsilon}(v) = \sum_{j=0}^s \sum_{k=0}^s \gamma_j \gamma_k R_{XX}(j+v-k).$$

Further, for positive v , $X(t)$ should be close to uncorrelated with $\epsilon(t+v)$, that is, $R_{X\epsilon}(v) = 0$, while for $v \leq 0$, we have

$$R_{X\epsilon}(v) = \sum_{j=0}^s \gamma_j R_{XX}(v-j).$$

Note that if we know only $R_{XX}(0), \dots, R_{XX}(M)$, then we can only calculate $R_{\epsilon\epsilon}(v)$ and $R_{X\epsilon}(-v)$ for $v = 0, \dots, M-s$.

Thus for given M and s we estimate the R_{XX} 's by sample autocovariances, then estimate the γ 's by the Yule-Walker equations, and then find the remaining elements of $\hat{\mathbf{A}}_1$ using the above expressions with estimates replacing the true values.

The ARMASEL Command

The command

ARMASEL(x, n, m, s, k1, k2, kopt, pval, p, q, alpha, beta, rvar, ier)

carries out the procedure described above. The first eight arguments are input and the last six are output although **alpha** and **beta** can be used as both input and output if **kopt** is negative as described below. The final model chosen has orders p and q returned in the integers **p** and **q**, coefficients α and β returned in the arrays **alpha** and **beta**, and noise variance returned in the real scalar **rvar**. The output integer **ier** is 0 if no errors are encountered, while it is 1 if the matrix being swept is judged to be singular at any step.

The arguments **x**, **n**, **m**, and **s** contain the data, the sample size, and M and s as described above. The arguments **k1**, **k2**, **kopt**, and **pval** allow the user to use **ARMASEL** in a variety of ways. First, the real scalar **pval** is the " p -value to enter" at each step of the stepwise procedure. Thus using a smaller value for

pval allows more variables to enter the model. Then **m**, **s**, **k1**, **k2**, and **kopt** (and possibly **alpha** and **beta**) together describe what lags are contenders for inclusion in the model. First, the largest AR or MA lag possible is **m-s** as described above. Then we have the following rules based on the value of **kopt**.

- kopt=0** - The **k1** AR lags (out of the possible **m-s**) having the largest $\hat{R}_{XX}(v)$ in absolute value and the **k2** MA lags having the largest $\hat{R}_{X\epsilon}(-v)$ in absolute value are contenders for inclusion.
- kopt=j>0** - This is the same as **kopt=0** except that the stepwise procedure continues until **j** lags are included.
- kopt=j<0** - The **k1** AR lags that are entered in **alpha** and the **k2** MA lags that are entered in **beta** are forced into the model. Thus **k1+k2= -j**.

The following code will use **ARMASEL** to estimate the coefficients of a full ARMA(*p*, *q*) model for user-specified **m** and **s** such that $m-s \geq \max(p, q)$:

```
alpha=LINE(p,0,1)
beta=LINE(q,0,1)
ppq=p+q
kopt=-ppq
ARMASEL(x,n,m,s,p,q,kopt,pval,p,q,alpha,beta,rvar,ier)
```

These estimators can be used as starting values for one of the more efficient estimation procedures that we have discussed (see the **SALESPRD.MAC** macro), or to calculate the AIC (see the **ARMAAIC.MAC** macro). The following code can be used to extract from the arrays **alpha** and **beta** the information needed by the **SEASEST** command if the user wants to produce asymptotically efficient estimators of the coefficients in the subset ARMA model as chosen by **ARMASEL**:

```
lags=<0>
coffs=<0>
lagsb=EXTRACT(beta,q,0,ne,nlagsb)
IF(nlagsb.eq.0,s1)
lags=<lagsb,lags>
coffsb=EXTRACT(beta,lagsb,nlagsb)
coffs=<coffsb,coffs>
;s1
lagsa=EXTRACT(alpha,p,0,ne,nlagsa)
IF(nlagsa.eq.0,s2)
lags=<lagsa,lags>
coffsa=EXTRACT(alpha,lagsa,nlagsa)
```

```

coeffs=<coeffsa,coeffs>
;s2
ords=<0,nlagsa,0,nlagsb,0>

```

Subset Autoregression

Subset AR models have been found to be particularly useful in representing some time series. For example, Tong (1977) found that for the log of the lynx data, an AR(11) model having only coefficients of lags 1, 2, 4, 10, and 11 nonzero has a value of AIC smaller than that of any full AR model (see Problem C3.14). The **ARMASEL** command can be used to fit such models.

Instead of looking for an AR model which has some zero coefficients, one alternative would be to look for an AR model which has some zero partial autocorrelations. Thus for data $X(1), \dots, X(n)$, one could calculate the partial autocorrelations $\hat{\theta}_1, \dots, \hat{\theta}_M$ for some M , then determine in some way a subset of these coefficients that are not significantly different from zero, and then use Levinson's algorithm backward to find the corresponding AR coefficients (see the **PARTAR** command).

One possibility for finding a subset of partials would be to start with no partials in the model, and then in a stepwise fashion add a partial, update the residual variance, and then find AIC for this subset. At each step the remaining partial having highest absolute value is the contender for addition, and the process stops when adding a partial would not decrease the AIC. The residual variance is easily updated since if we denote its value for k partials by $\hat{\sigma}_k^2$, we have from Levinson's algorithm

$$\hat{\sigma}_k^2 = \hat{\sigma}_{k-1}^2(1 - \hat{\theta}_{j_k}^2),$$

where $\hat{\theta}_{j_k}^2$ is the new partial being added.

Another possibility would be to include only partials that are larger in absolute value than some threshold value. One possibility for this threshold is $2/\sqrt{n}$ which is the threshold for partials for lag greater than p for an AR(p) process.

The DTAR Command

The command

```
alpha=DTAR(X,n,m,iop to,iop t1,iop t m,p,r0,rvar[,cat])
```

can be used to determine full and subset AR models according to a variety of rules. The array **X** and the integer **n** contain the data set to be analyzed and its length, respectively. Note that the sample mean should almost always be removed from **X** prior to calling **DTAR** as it does not do such a subtraction (it is

Table 3.7. Output of the DTAR Command for the Critical Radio Frequencies Data

i	part(i)	brv(i)	urv(i)	cat(i)
1	.893475	.201702	.202546	-4.916577
2	-.486187	.154024	.155319	-6.390981
3	.050179	.153636	.155581	-6.353332
4	.071237	.152857	.155448	-6.332052
5	-.062895	.152252	.155492	-6.303437
6	.249393	.142783	.146444	-6.672332
7	.140727	.139955	.144160	-6.751625
8	.205798	.134027	.138649	-6.997269
9	.209706	.128133	.133126	-7.265221
10	.327085	.114425	.119400	-8.093822
11	.201627	.109773	.115046	-8.374556
12	-.224861	.104223	.109708	-8.759499
13	-.370015	.089954	.095105	-10.115280
14	-.223356	.085466	.090760	-10.572720
15	-.009668	.085458	.091155	-10.479280
16	-.100213	.084600	.090643	-10.495350
17	-.126803	.083240	.089585	-10.579070
18	.048859	.083041	.089774	-10.509190
19	-.024806	.082990	.090125	-10.419610
20	-.058374	.082707	.090226	-10.360990

nord = 17

possible to imagine situations where one might want to perform the analyses in DTAR on data that have nonzero mean). The integer m contains the maximum AR order to allow, while the three integers $iopto$, $iopt1$, and $ioptm$ contain the user's choices about the order to use (1 means use order m while 2 means use an order-determining criterion), whether or not to display a table of partials and biased and unbiased residual variances and the CAT criterion for orders 1 through M (1 means yes and 2 means no), and finally which algorithm to use. Note that $ioptm$ can be 1, 2, 3, 4, 5, or 6. For 1, 3, and 5, the algorithm described in Theorem 3.4.5 is used, while for 2, 4, and 6, the Burg algorithm is used. Choices 1 and 2 mean to fit a full AR model of either order m or that chosen by CAT depending on whether $iopto$ is 1 or 2. Choices 3 and 4 mean to use AIC to find an AR model corresponding to a subset of partials as found by AIC, while choices 5 and 6 mean to use the $2/\sqrt{n}$ threshold to find such a subset of partials. In any case, the output variables p , α , and $rvar$ contain the order, coefficients, and error variance of the output model. The output real scalar $r0$ contains the sample variance, and the optional output argument cat contains the values of the CAT criterion for orders 1 through m . To illustrate the use of DTAR, consider Table 3.7 where we display the output of applying it to the critical radio frequencies data. The CAT criterion chooses order 17 (as did the AIC), and the value for order 14 is just slightly higher than that for order 17.

3.5.4. A Macro for Model Identification

We have seen in this section a wide variety of methods for choosing models for data. Because there are so many methods, the problem of model identification is very difficult. Thus we have included a macro with TIMESLAB that will lead a user through many of the methods that we have described. This macro is called `ID.MAC`. A listing of it is included at the end of this chapter (see Example 3.16). The macro introduces each part of the identification process with a screen describing what the user will see and what possible action should be taken in response to the nature of the diagnostics. Readers are urged to use this macro for a data set of their choice while looking at the listing of the macro. This will give experience both in identifying models and in how such a macro is written.

3.6. Box-Jenkins Modeling and Forecasting

The most popular ARMA-model based forecasting method is one proposed by Box and Jenkins (1970). The method uses the traditional statistical modeling strategy of (1) model identification, (2) parameter estimation, (3) diagnostic checking, and (4) consideration of alternative models, if necessary, within the class of multiplicative seasonal ARIMA models (see Section 2.5.7) to obtain a model suitable for forecasting. In this section we describe the various steps in the method and illustrate the TIMESLAB commands that can be used to produce Box-Jenkins forecasts. Note that the method uses only some of the procedures that we have described so far in the book. For example, most computer implementations of the method use none of the automatic model determination procedures. Also, it is only concerned with methods that are done in the time domain. The appeal of the method is its simplicity and the wide availability of computer software for using it.

The Models

The Box-Jenkins method for data $X(1), \dots, X(n)$ assumes that if we transform X to a series W by

$$W(t) = (1 - L)^d (1 - L^S)^D Y(t),$$

where

$$Y(t) = \begin{cases} (X(t) + m)^\lambda, & \lambda \neq 0 \\ \log(X(t) + m), & \lambda = 0, \end{cases}$$

and m is chosen to make $X(t) + m$ positive for all t (unless $\lambda = 1$), then a model of the form

$$\left(\sum_{j=0}^p \alpha_j L^j \right) \left(\sum_{k=0}^P \theta_k L^{kS} \right) (W(t) - \mu) = \left(\sum_{l=0}^q \beta_l L^l \right) \left(\sum_{r=0}^Q \gamma_r L^{rS} \right) \epsilon(t),$$

with $\epsilon \sim \text{WN}(\sigma^2)$, will adequately represent W . Note that this model for W can be written as an $\text{ARMA}(p+PS, q+QS)$ model by multiplying the polynomials in the model. One of the appeals of the Box-Jenkins models is that many of the coefficients in this general ARMA model are allowed to be zero. Thus this class of models allow us to represent data parsimoniously, that is, with a small number of parameters. This is the same idea as in the stepwise ARMA modeling embodied in the `ARMASEL` command (see Section 3.5.3).

The class of models for W is called the multiplicative seasonal ARMA model with orders $\langle p, P, q, Q, d, D, S \rangle$ and coefficients $\langle \alpha, \theta, \beta, \gamma \rangle$. In this book we will generalize this model slightly by removing the restriction that the powers of L in the second and fourth polynomials above are of the form kS and mS . This allows us to include the general subset ARMA model in the class of models that we can consider. Thus we have the model

$$g(L)G(L)[W(t) - \mu] = h(L)H(L)\epsilon(t),$$

where

$$\begin{aligned} g(L) &= \sum_{j=0}^p \alpha_j L^j, & G(L) &= \sum_{k=0}^P \theta_k L^{u_k}, \\ h(L) &= \sum_{l=0}^q \beta_l L^l, & H(L) &= \sum_{m=0}^Q \gamma_m L^{v_m}, \end{aligned}$$

are called the full and subset AR and full and subset MA operators, respectively. Each of these is assumed to have all of their zeros outside the unit circle, so that the model is stationary and invertible. We can also write the model with a constant term τ_0 , that is,

$$g(L)G(L)W(t) = \tau_0 + h(L)H(L)\epsilon(t),$$

where

$$\tau_0 = g(1)G(1)\mu = \left(\sum_{j=0}^p \alpha_j \right) \left(\sum_{k=0}^P \theta_k \right) \mu.$$

From our previous discussions, it should be clear that the Y transform is used to stabilize variance, the W transform removes trends and cycles, and the model for W represents the behavior in the data after this “preprocessing.” In the remainder of this section we describe and illustrate the four steps in the Box-Jenkins procedure.

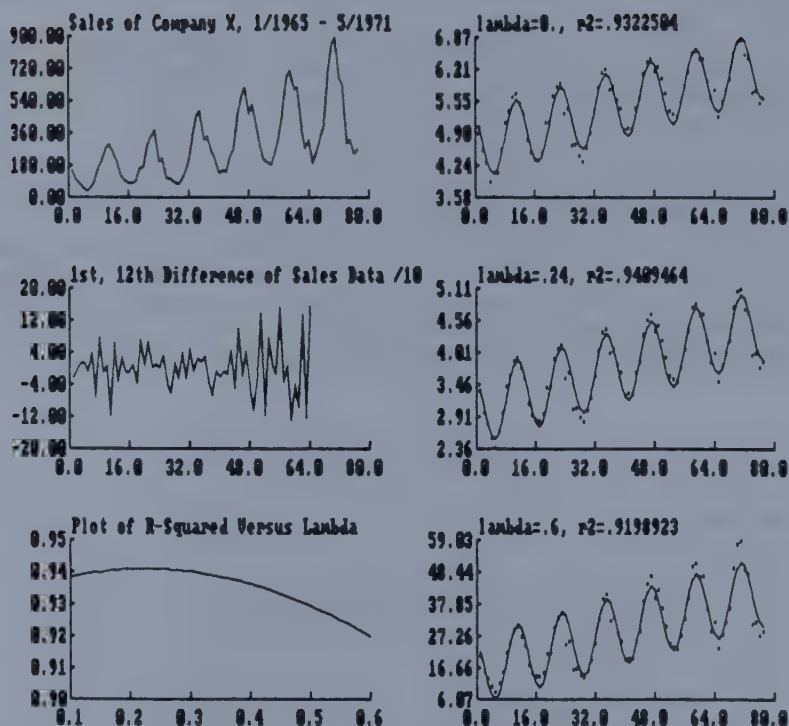


Figure 3.10. Example of Identifying λ .

Model Identification

The model identification step itself consists of three parts:

1. Determine if the power transform from X to Y is required, and if so, what values of m and λ should be used. Note that m is only included to make $X(t) + m$ positive for all t .
2. Determine if the differencing transform from Y to W is necessary, and if so, what values of d , D , and S should be used.
3. Determine if an ARMA model is required to model W , and if so, what values of the orders and lags should be used.

Identifying λ

This step is used either (1) to stabilize the variance of X or (2) to produce a time series Y for which the differencing step can eliminate any trends and/or cycles. This step is not needed for any series whose variance appears constant and which has no trends or cycles. To illustrate a case where the power transform is needed, consider the series in Figure 3.10, which consists of the sales of an industrial company for 77 months (see Chatfield and Prothero (1973) and Box and Jenkins (1973)). This data set consists mainly of a linear trend and an annual cycle whose amplitude is increasing with time at a faster than linear rate. If no power transformation is applied, no amount of differencing can remove the cycle. For example, we have included in Figure 3.10 a graph of the first and 12th difference of X . Since all of the data are positive, we let $m = 0$ and try to find a suitable value for λ .

There are a variety of possible methods for selecting λ . Unfortunately, which method to use is somewhat subjective and depends on the nature of the data. The following macro tries a variety of λ 's and chooses the one that has the highest value of R^2 for the regression model

$$X^\lambda(t) = a + bt + c \cos \frac{2\pi(t+1)}{12} + \epsilon(t),$$

that is, the value that leads to the series that is best fit by a linear trend plus a cosine curve. See Box and Cox (1964) for a general discussion of choosing power transformations in regression analysis. Note that the cosine is shifted two months because the data start in January and the maximum yearly value is in November.

```

1 ;;
2 ;;   LAMBDA.MAC: macro to choose the power transform for
3 ;;               the sales data.
4 ;;
5 ;;   INPUT: nlam, lam1, lam2 (number of powers and first
6 ;;               and last powers to use)
7 ;;
8 ;;   OUTPUT: r2s, lam (arrays of R-Squared and lambda's)
9 ;;
10 PAUSE
11 ;start
12 READ(sales,x,n)
13 ;start1
14 PLOTOM
15 lam=LINE(nlam,lam1,lam2,1) ;lam is array of lambda's
16 k=3
17 npk=n+k
18 cs=COS(npk,1,12)
19 cs=EXTRACT(cs,k,npk)      ;shift cosine by two units

```

```

20 ll=LINE(n,0,1)           ;form vector 1,...,n
21 one=LINE(n,1,0)          ;intercept
22 xx=<one,ll,cs>            ;design matrix
23 r2s=LINE(nlam,0,0)        ;reserve array to get R squares
24 i=1                       ;initialize loop
25 ;
26 ;startloop
27 ;
28 lambda=lam[i]
29 if(lambda.gt..001,nolog) ;check if we need to do log
30 y=LOGE(x,n)               ;yes
31 GOTO(reg)
32 ;nolog                    ;no
33 y=x^lambda
34 ;reg
35 yy=SUBMWS(y,n,1,ybar)     ;find sum of squares of transformed data
36 ssy=DOT(yy,yy,n)
37 REG(y,xx,n,3,beta,rss,t,res) ;do regression
38 sse=dot(res,res,n)        ;find R squared
39 r2s[i]=1-{sse/ssy}
40 yhat=y-res                ;superimpose data and fitted values
41 LABEL(y)='lambda=@lambda@, r2=@r2s[i]@'
42 LABEL(yhat)=' '
43 CLS
44 PLOT2(y,yhat,n,n,2,1)
45 IF(i.eq.nlam,endloop)     ;check if loop finished
46 i=i+1                     ;no
47 GOTO(startloop)
48 ;
49 ;endloop                  ;yes
50 ;
51 PLOTOFF

```

In Figure 3.10 we have included a graph of R^2 for 51 values of λ between .1 and .6, and the graphs of X^λ for $\lambda = 0$, $\lambda = .24$ (the value giving the maximum R^2), and $\lambda = .6$. The log transform is ruled out because of how poorly the first cycle of the cosine fits the data.

The method of the previous paragraph is appropriate for these data because the transformed data do seem to be sinusoidal. In general, one chooses the transformation so that the resulting data most closely match some deterministic function that can be removed by differencing.

Identifying Differences

In this part of the process, one looks for the least amount of differencing required to produce a correlogram and partial correlogram that can be matched by that of a multiplicative subset ARMA process having only a few parameters. This is not always an easy process as it requires rather extensive experience. In Section 3.4 we described a variety of types of correlograms and

partial correlograms and the corresponding models. Unfortunately no such description can be exhaustive. Note that doing more differencing leads to losing observations, while doing less may lead to having to use more parameters in the ARMA model. The probability limits for prediction that we will obtain below are a function of the number of observations minus the number of estimated parameters. Thus there is a tradeoff in degrees of freedom in these two parts of the identification process.

To illustrate differencing, consider again the sales data. In Table 3.8, we give the values of the correlogram and partial correlogram for the power transformed data and for its first, 12th, and first and 12th differences.

Table 3.8. Table of Correlations and Partial Correlations for Various Differenced Versions of the Power Transformed Sales Data

```

Correlations of Y
  1 | .89 .71 .45 .19 -.01 -.09 -.07 .08 .28 .48 .63 .69
13 | .61 .46 .25 .03 -.14 -.21 -.19 -.08 .07 .24 .37 .42
25 | .38 .26 .09 -.10 -.26 -.34 -.36 -.29 -.18 -.04 .07 .14

Partial of Y
  1 | .89 -.42 -.50 .08 .29 .43 .12 .15 .13 -.06 .03 -.04
13 | -.28 .04 .13 -.02 -.18 -.04 .11 -.05 -.08 .12 -.07 .05
25 | -.06 -.05 -.10 -.11 .02 .02 -.11 -.08 -.08 .09 -.05 -.02

Correlations of 1st Difference
  1 | .33 .40 -.07 -.29 -.57 -.49 -.54 -.25 .02 .24 .45 .61
13 | .36 .27 -.01 -.23 -.44 -.43 -.43 -.20 -.05 .24 .27 .50
25 | .29 .29 .04 -.13 -.35 -.34 -.35 -.20 -.10 .15 .18 .39

Partial of 1st Difference
  1 | .33 .33 -.33 -.45 -.46 -.13 -.20 -.24 -.03 -.06 -.04 .24
13 | -.08 -.16 .06 .19 -.03 -.14 .04 .13 -.16 .01 -.15 .05
25 | .08 .05 .00 -.08 .02 .11 .05 .04 -.14 .07 .01 -.01

Correlations of 12th Difference
  1 | .40 .54 .23 .19 .07 .04 -.13 -.10 -.06 -.14 .08 -.22
13 | -.10 -.22 -.14 -.12 -.06 -.10 -.04 -.03 -.02 -.06 -.14 -.20
25 | -.17 -.08 -.09 .01 .00 .06 .10 .10 .00 -.01 -.13 -.14

Partial of 12th Difference
  1 | .40 .45 -.10 -.12 -.02 .01 -.20 -.05 .22 -.11 .16 -.30
13 | -.17 .03 .03 .10 .00 -.01 -.08 -.11 .03 -.23 .02 -.11
25 | .02 .22 -.17 .07 .00 -.06 .05 -.13 .01 -.15 -.03 -.16

Correlations of 1st,12th Difference
  1 | -.60 .36 -.22 .06 -.08 .11 -.17 -.02 .12 -.26 .44 -.35
13 | .18 -.14 .05 -.04 .09 -.09 .05 .00 .03 .04 -.02 -.08
25 | -.05 .09 -.11 .10 -.05 .02 .04 .08 -.07 .08 -.08 -.08

Partial of 1st,12th Difference
  1 | -.60 .01 .00 -.10 -.13 .06 -.10 -.31 .04 -.22 .23 .05
13 | -.16 -.12 -.15 -.05 -.06 .01 .05 -.11 .14 -.09 .02 -.11
25 | -.29 .11 -.12 -.04 .01 -.10 .09 -.08 .08 -.04 .05 -.05

r0x=38863.330000
r0=.397435
r01=.083808
r012=.036166
r0112=.044060

```

Identifying ARMA Models

Once the data are differenced (if necessary), the next part of the method is to find an ARMA model that adequately represents the differenced data. Here only the sample autocorrelation and partial autocorrelation functions are used. To illustrate this process, we continue with a discussion of the sales data. Note that the correlogram and partial correlogram of the first and 12th difference series have an obvious pattern that the others don't. First, the lag one partial is large and is followed by several that are small, followed by an increase at lags 10 and 11. The first three correlations ($-.6$, $.36$, $-.22$) are decaying exponentially with alternating signs, indicating the presence of an AR(1) term in the model with coefficient approximately $.6$. The presence of large correlations at lags 10, 11, and 12 indicates the presence of a subset MA term in the data. Since the maximum correlation is at lag 11, it is tempting to conclude that the MA term should be of lag 11. The macro given below finds the true autocorrelation function for the best fitting models having an AR(1) term and then either an MA(11) term or an MA(12) term. The output of the macro is given in Table 3.9.

Table 3.9. Output of the SEASEST Macro

```

Coefficients From SEASEST
  1 |      .504522      .780130
rvar=.025497
Correlations for AR(1),MA(11)
  1 | -.50   .25  -.13   .06  -.02   .00   .02  -.06   .12  -.24   .48  -.24
 13 |  .12  -.06   .03  -.02   .01   .00   .00   .00   .00   .00   .00   .00
Coefficients From SEASEST
  1 |      .501941      -.801871
rvar=.024118
Correlations for AR(1),MA(12)
  1 | -.50   .25  -.13   .06  -.03   .01   .01  -.03   .06  -.12   .24  -.49
 13 |  .24  -.12   .06  -.03   .02  -.01   .00   .00   .00   .00   .00   .00

```

The residual variance for the model having the MA(12) term is smaller than that for the model having the MA(11) term but the correlations for the latter model appear to match those of the data better. There seems to be no clear optimal choice between the two models.

```

1 ;;
2 ;;   SALESRHO.MAC: macro to find the correlation function of
3 ;;                               the best fitting model with AR(1),MA(11)
4 ;;                               and AR(1),MA(12) for the 1st and 12th
5 ;;                               difference of the fourth root of the sales data.
6 ;;
7 PAUSE
8 ;start

```



```

9 ;
10 ;   Read and transform data:
11 ;
12 READ(sales,x,n)
13 ;start1
14 x=x~.25
15 x=DIFF(n,1,x)
16 nm1=n-1
17 x=DIFF(nm1,12,x)
18 nm13=n-13
19 ;
20 ;   AR(1),MA(11):
21 ;
22 ords=<1,0,0,1,0>
23 lags=<11>
24 coeffs=<0,0>
25 SEASEST(x,nm13,ords,coeffs,lags,20,.001,20,rvar,ier,sds)
26 LIST(coeffs,2)
27 LIST(rvar)
28 alpha=<coeffs[1]>
29 beta=LINE(11,0,0)
30 beta[11]=coeffs[2]
31 rho=ARMACORR(alpha,beta,1,11,rvar,24,r0,ier)
32 LABEL(rho)='Correlations for AR(1),MA(11)'
33 LIST(rho,24,12,12f5.2)
34 ;
35 ;   AR(1),MA(12):
36 ;
37 lags=<12>
38 coeffs=<0,0>
39 SEASEST(x,nm13,ords,coeffs,lags,20,.001,20,rvar,ier,sds)
40 LIST(coeffs,2)
41 LIST(rvar)
42 alpha=<coeffs[1]>
43 beta=LINE(12,0,0)
44 beta[12]=coeffs[2]
45 rho=ARMACORR(alpha,beta,1,12,rvar,24,r0,ier)
46 LABEL(rho)='Correlations for AR(1),MA(12)'
47 LIST(rho,24,12,12f5.2)

```

Estimation and Diagnostic Checking

Once a model has been determined, the approximate likelihood estimation procedure is used to estimate its parameters (see the **SEASEST** command), while the portmanteau test is used to determine if the residuals of the fitted model are white noise (see the **QTEST** command).

Forecasting

Given estimates of the parameters of a model that has been judged to be adequate, the Box-Jenkins procedure then produces forecasts as follows. For simplicity we suppose that $\mu = 0$. Then we have

$$g(L)G(L)(1-L)^d(1-L^S)^DY(t) = h(L)H(L)\epsilon(t),$$

where Y is the power transformed series. The result of using the **SEASEST** command is estimates of the coefficients of the four polynomials g, G, h , and H , that is, the parameters of the model for

$$W(t) = (1-L)^d(1-L^S)^DY(t).$$

This model for W can be translated into a model for Y , and thus we can write Y as an ARMA model of orders $p^* = p + u_P + d + SD$ and $q^* = q + v_Q$. Note that if differencing was done, then some of the zeros of the polynomial for the AR part of the model for Y will be on the unit circle.

Now given this ARMA(p^*, q^*) model for Y , a recursion for the forecasts of Y can be used. This recursion is similar to the ones used in the approximate MLE procedure. Once forecasts for Y are obtained they can be converted to forecasts for the original series X by doing the transformation that is the inverse of the one done in the power transform step.

The SEASPREDD Command

The command

```
SEASPREDD(X,n,ords,coeffs,lags,rvar,tf,t1,h1,conf,
xp,xpl,xpu,ier)
```

will carry out the forecast stage for any model that can be expressed in the general multiplicative seasonal ARMA framework that we have described in this book, including the Box-Jenkins models. All of the arguments of the command except the last four are input. The array X and the integer n contain the original data and sample size. Then we have

$$\text{ords} = \langle p, P, q, Q, M, d, D, S \rangle,$$

lags contains the lags for the subset AR and MA models, while **coeffs** contains the values for the full and subset AR coefficients followed by the coefficients of the full and subset MA coefficients, followed by the value of the constant if there is one, followed finally by the values of m and λ for the power transform. The scalar **rvar** contains the value of the error variance. Then the integers **tf** and **t1** contain the range of prediction origins to use, and the integer **h1** contains the largest number of steps ahead to forecast. Thus for each value of

t included in the range tf to tl , SEASPRED will use $X(1), \dots, X(t)$ to forecast $X(t+1), \dots, X(t+hl)$. These forecasts are put into the output array xp with the hl forecasts starting at $t = tf$ coming first, then the ones for $t = tf + 1$, and so on. The output arrays xpl and xpu contain lower and upper probability limits (using the confidence level specified by the user in the real scalar $conf$) for the corresponding elements of xp . Finally, the output integer ier is 0 if no error was encountered or 1 if an illegal power transform is specified.

To illustrate the use of SEASPRED, consider Figure 3.11 which contains the sales data with forecasts of the next 24 values of the series and 95% probability limits appended. The macro that produced this graph is given below. Note that we used the model with the MA(12) term in the macro, and that in order to obtain a graph with satisfactory tic mark labels on the vertical axis, we divided the values being plotted by 10.

```

1 ;;
2 ;;   SALESPRD.MAC: macro to produce forecasts of the next
3 ;;                               24 values of the sales data and to construct
4 ;;                               a plot of the last 36 values followed by the
5 ;;                               forecasts and probability limits.
6 ;;
7 ;;   INPUT: none
8 ;;
9 PAUSE
10 ;start
11 READ(sales,x,n)
12 ;
13 ;   power transform and 1st and 12th differences:
14 ;
15 xx=x^.25
16 xx=DIFF(n,1,xx)
17 nm1=n-1
18 xx=DIFF(nm1,12,xx)
19 nm13=n-13
20 ;
21 ;   use ARMASEL to get initial estimates:
22 ;
23 alpha=<1>
24 beta=<12>
25 ARMASEL(xx,nm13,40,20,1,1,-2,.95,p,q,alpha,beta,rvar,ier)
26 ;
27 ;   use SEASEST to find approximate MLE's:
28 ;
29 ords=<1,0,0,1,0>
30 coeffs=<alpha[1],beta[12]>
31 lags=<12>
32 SEASEST(xx,nm13,ords,coeffs,lags,20,.001,20,rvar,ier,sds)
33 LIST(coeffs,2)
34 ;
35 ;   use SEASPRED to get forecasts:
36 ;

```

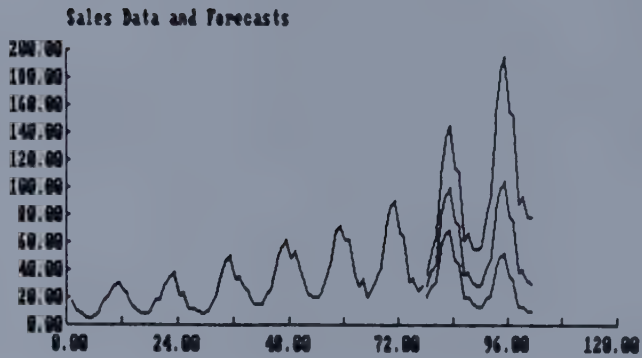


Figure 3.11. Sales Data and Forecasts with 95% Probability Limits.

```

37 ords=<1,0,0,1,0,1,1,12>
38 lags=<12>
39 coeffs=<coeffs,0,.25>
40 tf=n
41 tl=n
42 hl=24
43 conf=.95
44 SEASPRE(x,n,ords,coeffs,lags,rvar,tf,tl,hl,conf,xp,xpl,xpu,ier)
45 ;
46 ; The rest of the macro constructs the plot:
47 ; For the vertical scale to work properly, I have to
48 ; divide the values by 10.
49 ;
50 np1=n+1
51 np24=n+24
52 x1=LINE(24,n,1)
53 x2=LINE(n,0,1)
54 yy=<x, xp, xpl, xpu>
55 xx=<x2, x1, x1, x1>
56 yy=yy/10.
57 LABEL(yy)='Sales Data and Forecasts'
58 LABEL(xx)=' '
59 nn=<n,24,24,24>
60 type=<2,2,2,2>
61 ;
62 ; Plot: I got the scale values by looking at an unscaled plot
63 ; and choosing values giving good tic mark labels.
64 ;
65 PLOT(x,yy,nn,4,type,0,120,0,200)

```

3.7. Other Modeling Strategies

An important feature of the Box-Jenkins modeling and forecasting procedure is that the initial detrending and deseasonalizing transformations are done by differencing and not by regressing the data on deterministic functions of time.

3.7.1. Regression with Autoregressive Errors

Suppose we would like to model $X(t)$ as a deterministic function of time and the errors appear to be autocorrelated; that is, we have

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{y} = (X(1), \dots, X(n))^T$, \mathbf{X} is an $(n \times m)$ matrix of fixed constants, and $\boldsymbol{\epsilon} = (\epsilon(1), \dots, \epsilon(n))^T$ has covariance matrix $\sigma^2 \mathbf{V}$. In this section we assume that $\boldsymbol{\epsilon}$ can be adequately modeled as an $\text{AR}(p)$ process for some order p . We can estimate the parameters by the following iterative procedure.

1. Use ordinary least squares to find initial estimates $\hat{\boldsymbol{\beta}}_0$ and residuals \mathbf{e}_0 .
2. Determine the order \hat{p}_1 , coefficients $\hat{\boldsymbol{\alpha}}_1$, and error variance $\hat{\gamma}_1^2$ of an autoregressive model for \mathbf{e}_0 .
3. Create a new observation vector \mathbf{z}_1 and regression matrix \mathbf{W}_1 by applying the AR filter found in step 2 to \mathbf{y} and \mathbf{X} . This is done using the `ARFILT` command (see Section 2.6).
4. Now apply ordinary least squares to \mathbf{z}_1 and \mathbf{W}_1 to obtain the coefficient estimate $\hat{\boldsymbol{\beta}}_1$ and residuals $\mathbf{e}_1 = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_1$.
5. Return to step 2 with \mathbf{e}_1 replacing \mathbf{e}_0 .

This process continues until successive iterations result in the same value of the AR order. The successive values of the quantities involved will have subscripts 1, 2, ... The following macro will carry out this procedure. Note that it also calculates the covariance matrix of the final regression coefficient estimates, as well as the residuals for the final AR process.

```

1 ;;
2 ;;   REGAR.MAC: This macro does regression with AR errors, i.e.
3 ;;
4 ;;
5 ;;       y = X beta + epsilon
6 ;;
7 ;;       where X is n x m, and epsilon is N(0,s2*V)

```

```

8 ;;
9 ;;
10 ;;  INPUT: y, X, n, m
11 ;;          maxp (maximum AR order)
12 ;;
13 ;;  OUTPUT: beta, p (AR order), resar (residuals of AR)
14 ;;          V (covariance matrix of beta)
15 ;;
16 PAUSE
17 ;start
18 z=y          ;we'll use z and W for transformed y and X
19 W=X
20 p=0
21 ;
22 ;startloop
23 ;
24 PROMPTOFF
25 REG(z,W,n,m,beta,rss,t,res)      ;do regression (first time is OLS)
26 yy=MMULT(X,beta,n,m,1)          ;find residuals of regression model
27 res=y-yy
28 rho=CORR(res,n,maxp,0,1,r0,per)  ;fit AR to residuals
29 rvar=r0                          ;
30 alpha=CORRAR(rho,r0,maxp,n,p1,rvar,cat) ;
31 list(p1)
32 IF(p1.eq.p,endlloop)             ;check if same order as last
33 p=p1
34 IF(p.eq.0,endlloop)
35 z=ARFILT(alpha,p,rvar,y,n,1,ier) ;filter y and X using new AR
36 W=ARFILT(alpha,p,rvar,X,n,m,ier)
37 resar=ARFILT(alpha,p,rvar,res,n,1,ier) ;resar is residuals of AR
38 GOTO(startloop)
39 ;
40 ;endlloop
41 ;
42 V=MMULT(W,W,n,m)                ;now find covariance matrix of final beta
43 V=MINV(V,m,ier)
44 V=rvar*V
45 PROMPTON

```

As we will see in the next section, one possible model for the log (base 10) of the lynx data is

$$x(t) = \mu + a \cos \frac{2\pi t}{9.5} + b \sin \frac{2\pi t}{9.5} + \epsilon(t),$$

where ϵ is an $AR(p)$. The next macro uses the REGAR macro and finds $p = 2$ with coefficients -1.037 and 0.351 , while the estimates of the regression coefficients are 2.901 , 0.354 , and -0.469 .


```

1 ;;
2 ;;   LREGAR.MAC: macro to do regression with AR errors
3 ;;               for log10 of lynx data.
4 ;;
5 ;;
6 ;;       y(t)= mu + a cos(2pi t/per) + b sin(2pi t/per) + eps(tx)
7 ;;
8 ;;
9 ;;   INPUT:   per, maxp (maximum AR order)
10 ;;
11 PAUSE
12 ;start
13 READ(lynx,x,n)
14 ;start1                      ;entry point if we already read data
15 y=LOGE(x,n)
16 ee=LOGE(10)
17 y=y/ee
18 const=LINE(n,1,0)
19 np1=n+1
20 c1=COS(np1,1,per)
21 c1=EXTRACT(c1,2,np1)          ;have to do this so 1st element is cos(2pi/per)
22 s1=SIN(np1,1,per)
23 s1=EXTRACT(s1,2,np1)
24 X=<const,c1,s1>               ;X is now design matrix for regression
25 m=3
26 MACRO(regar,start)           ;let REGAR.MAC do the work

```

Intervention Analysis

An important use of regression with autocorrelated errors is the situation where it is suspected that some outside force causes a shift in the level of a time series at some time point t_0 . This force is referred to as an intervention and the method of analysis is called intervention analysis (see Box and Tiao (1975)). The basic idea of intervention analysis is that the intervention affects the mean of the time series in a way that can be modeled as a deterministic function of time, and that the correlation structure is not effected. The simplest example of this is the case where the mean of the series is a constant μ_1 before the intervention and a constant μ_2 afterward, and we want to test whether $\mu_1 = \mu_2$. If the correlation structure of the series can be adequately represented by an autoregressive process, then we can use the REGAR macro described above to estimate μ_1 and μ_2 by having the design matrix consist of two columns, the first consisting of 1's up to and including the intervention followed by 0's, and the second being 0's up to and including the intervention followed by 1's. Then we can use the results of Theorem A.5.4 to test the hypothesis that $\mu_1 = \mu_2$.

Other models of the effect of an intervention are often of the form of an abrupt increase or decrease in the mean followed by a decay back to the original mean. Any effect that can be modeled as a deterministic function of time can be put into the framework of a regression with autocorrelated errors.

3.7.2. ARARMA Modeling

One very simple method for automatically modeling and forecasting data has been suggested by Parzen (1982) and used successfully by Newton and Parzen in a forecasting competition organized by Makridakis (see Makridakis et al. (1984)). The method consists of two parts:

1. For some maximum lag M_1 , calculate the regression coefficients

$$\hat{\beta}_k = -\frac{\sum_{t=1}^{n-k} X(t)X(t+k)}{\sum_{j=1}^n X^2(t)}, \quad k = 1, \dots, M_1.$$

Then let m be the value of k having the largest $|\hat{\beta}_k|$, and form

$$e(t) = X(t+m) + \hat{\beta}_m X(t), \quad t = 1, \dots, n-m.$$

2. Fit an autoregressive process to the e 's, using maximum possible order M_2 and an order-determining criterion to determine the order p to use. In the general ARARMA procedure one would fit an ARMA model to the e 's, but in most cases an AR process is adequate.

The first part is called the first AR since it is in essence fitting a one lag subset AR model to X . The second part is called the second AR. Note that the first AR may result in a coefficient that is greater than one, while in the second AR, it is recommended that the Yule-Walker or Burg estimators be used to guarantee that the process fit to the e 's is stationary.

The result of this procedure is a model of the form

$$(1 + \hat{\beta}_m L^m) \hat{g}(L) X(t) = \epsilon(t),$$

where \hat{g} is the AR operator determined in part 2 of the procedure. This model is similar in form to the Box-Jenkins model (with no MA terms) except that it is easily made automatic, and the data determine the nature of the first AR, which is analogous to differencing and is in fact sometimes referred to as "quasi-differencing". If the first AR turns out to be stable, that is, the coefficient has absolute value less than one, then the forecasts will eventually converge to the mean of the observed data. As we discussed in Section 1.6.4, forecasts that follow a difference equation of necessity either are explosive or must converge to some finite value. In the short run this may not be troublesome, but if the analyst has a feel for the long-run nature of forecasts, then this information should be incorporated into whatever model is used, either by modeling this

behavior using a regression model, or by using differencing if polynomial growth is expected, or by insisting on a stable ARMA model if the series is expected to remain fairly constant.

The DTFORE Command

For a data set entered in the array **x** whose length is entered in the integer **n** and input integers **M1**, **M2** (which correspond to M_1 and M_2 above), and **npreds**, the command

```
DTFORE(x,n,M1,M2,npreds,m,p,beta,alpha,xp)
```

will fit the model described above, and will return the chosen lag m and order p in the integers **m** and **p**, as well as the array **xp** of length **n+npreds** which contains the first **m+p** values of **x**, followed by the one step ahead predictors of **x(m+p+1)** through **x(n)**, followed finally by forecasts of the next **npreds** values of **x**.

The Sales Data Again

To illustrate the simplicity of the DTFORE command, the macro DTFORE.MAC given below was used to find for the fourth root of the sales data that $m=12$, $p=2$, $\text{beta}=-1.0566$, and $\text{alpha}=-.229, -.457$. The first AR essentially chose 12th differences, except that the coefficient is actually more nonstationary than the -1 that would mean twelfth difference. In data such as this, this quasi-differencing will remove most of the variation in the data, and at least in the short term future, the model used in the second part of the model fitting has little impact on the nature of the forecast. Note further in this example the quasi-differencing will ultimately lead to more explosive growth than the actual 12th difference.

In Figure 3.12, we display the sales data, the values fitted by the model (the actual data are represented by the \times 's), and the forecasts of the next 24 values of the series. The values to be plotted were again divided by 10 prior to calling the PLOT2 command to superimpose the plots of the data and forecasts. Thus this figure is comparable to the one using the Box-Jenkins method.

```
1 ;;
2 ;;   DTFORE.MAC: macro to illustrate the use of the DTFORE
3 ;;               command by finding forecasts of the next 24
4 ;;               values of the sales data.
5 ;;
6 ;;   INPUT: none
7 ;;
8 PAUSE
9 ;start
10 READ(sales,x,n)
```

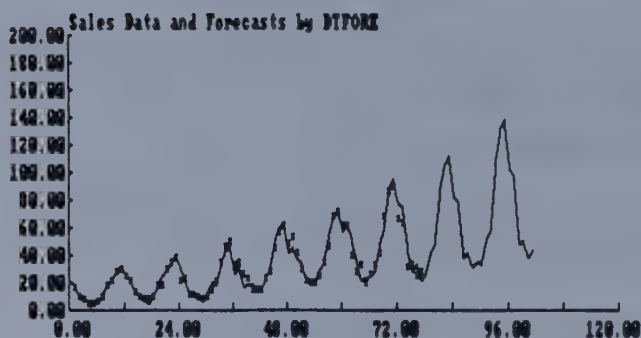


Figure 3.12. Sales Data and Forecasts As Determined by DTFORE.

```

11 x=x*.25           ;find fourth root
12 DTFORE(x,n,12,15,24,lag,p,alpha1,alpha2,xp)
13 LIST(lag,p)
14 LIST(alpha1)
15 LIST(alpha2,p)
16 x=x^4.           ;reconstruct original data
17 xp=xp^4.
18 x=x/10.
19 xp=xp/10.
20 LABEL(x)='Sales Data and Forecasts'
21 LABEL(xp)=' '
22 PLOT2(x,xp,n,101,2,1,0,120,0,200)

```

If the first AR chosen by DTFORE is stable, then the SEASPREDD command can be used to find probability limits for the forecasts. The intent of DTFORE however is to provide an easy-to-use command that gives forecasts that have been shown (in the forecasting competition) to compare very favorably with those given by more elaborate methods.

3.7.3. State Space and Bayesian Forecasting

In the recent time series literature, much attention has been devoted to analyzing data through the use of models that take on the structure that is used by the Kalman Filter Algorithm for generating forecasts (see Section A.4.3). Thus one models the “state” of a time series at time t recursively in terms of the state at the previous time. If one interprets this state space model from a Bayesian point of view (see Harrison and Stevens (1976)), then one can generate what are called Bayesian forecasts. The state space formulation has been used extensively in a number of other contexts as well.

3.8. Parametric Spectral Density Estimation

In Section 3.2 we considered nonparametric methods for estimating the spectral density of a covariance stationary time series. In that section we saw that (1) the choice of the scale parameter M is the key to finding a good estimator, (2) it is difficult to produce an estimator having sharp peaks without introducing possibly spurious peaks as well, and (3) the fastest that the expected mean square error can go to zero for a window estimator having lag window generator with characteristic coefficient q is $n^{-2q/(2q+1)}$, no matter how smooth is the true spectral density (as long as f is p -differentiable for some $p \geq q$). In this section, we consider using the spectral density of an ARMA model as an approximation to that of the process being analyzed. We will see that this parametric spectral density estimation procedure is easy to implement and under a wide variety of conditions leads to estimators superior to window estimators.

3.8.1. Autoregressive Spectral Estimation

In this section we assume that the process being observed can be written as an infinite order autoregressive process. As we discussed in Chapter 2, this includes the cases of a finite order autoregressive process and an invertible MA or ARMA process. In the next theorem (due to Masani (1966)), we give a condition for a process to be expressible as an infinite order AR process.

Theorem 3.8.1

CONDITIONS FOR $AR(\infty)$ REPRESENTATION

If X is a covariance stationary time series having a spectral density satisfying

$$0 < \lambda_1 \leq f(\omega) \leq \lambda_2 < \infty, \quad \omega \in [0, 1],$$

for some constants λ_1 and λ_2 , then X can be written as an infinite order AR process.

All ARMA processes except those that have a zero of the moving average polynomial on the unit circle have spectral densities that satisfy the conditions of this theorem. We consider approximating the true spectral density f of a process that can be written as an infinite order autoregression by that of a p th order AR process. Given data $X(1), \dots, X(n)$, autoregressive spectral estimation consists of three steps: (1) determining the order k of the best approximating AR spectral density f_k , (2) finding estimates $\hat{\alpha}_k(1), \dots, \hat{\alpha}_k(k)$

and $\hat{\sigma}_k^2$ of the parameters of f_k , and (3) estimating f by

$$\hat{f}_k(\omega) = \frac{\hat{\sigma}_k^2}{\left| \sum_{j=0}^k \hat{\alpha}_k(j) e^{2\pi i j \omega} \right|^2}.$$

To determine the order, we can use any of the methods described in Section 3.5. However, Parzen's CAT criterion was specifically proposed in the context of autoregressive spectral estimation, so we describe it here. Recall that the CAT criterion for order k is given by

$$\text{CAT}(k) = \begin{cases} \left(\frac{1}{n} \sum_{j=1}^k \frac{1}{\tilde{\sigma}_j^2} \right) - \frac{1}{\tilde{\sigma}_k^2}, & k = 1, \dots, M \\ - \left(1 + \frac{1}{n} \right) \hat{R}(0), & k = 0, \end{cases}$$

where $\tilde{\sigma}_j^2$ is the unbiased residual variance for order j . The basic motivation for the criterion is contained in the next theorem.

Theorem 3.8.2 PROPERTIES OF CAT

Let $\hat{\sigma}_p^2$ be the error variance and

$$\hat{g}_p(z) = \sum_{j=0}^p \hat{\alpha}_p(j) z^j$$

be the transfer function of the estimated AR process of order p based on a realization of length n from a process which can be written as an infinite order autoregressive process having coefficients $\alpha_1, \alpha_2, \dots$, transfer function

$$g_\infty(z) = \sum_{j=0}^{\infty} \alpha_j z^j,$$

and error variance σ_∞^2 . Let $\text{CAT}(p)$ be the CAT criterion for order p . Then

$$\lim_{n \rightarrow \infty} \text{E}(\text{CAT}(p)) = \lim_{n \rightarrow \infty} J_p,$$

where

$$J_p = \int_0^1 \text{E} \left(\left| \frac{1}{\hat{\sigma}_p^2} \hat{g}_p - \frac{1}{\sigma_\infty^2} g_\infty \right|^2 \right) f(\omega) d\omega.$$

This result shows that the CAT criterion chooses the order that results asymptotically in the spectral estimator that is closest (in the sense of the integrated relative mean square error measured by J_p) to the true $AR(\infty)$ transfer function. This is the origin of the name "criterion autoregressive transfer function."

To illustrate autoregressive spectral estimation, consider Figure 3.13 which contains the autoregressive spectral estimators (using CAT determined orders) for the three series whose nonparametric spectral estimators were given in Figure 3.8. Note how these estimators have fewer spurious "wiggles" than did the window estimators. This again shows that the AR method is more successful in resolving sharp peaks than is window estimation. A final note about the critical radio frequencies data is that the window estimator for truncation point 48 did not exhibit a clear peak at the sunspot cycle frequency, while the one for truncation point 96 did. A similar phenomenon occurs for autoregressive spectral estimation, namely that the estimate for order 14 does not have the low frequency peak (see Problem C3.24).

In the next theorem (due to Berk (1974)) we present the basic sampling properties of the autoregressive spectral estimator.

Theorem 3.8.3	PROPERTIES OF AR SPECTRAL ESTIMATION
----------------------	---

Let X be a covariance stationary time series that can be expressed as an infinite order autoregressive process

$$\sum_{j=0}^{\infty} \alpha_j X(t-j) = \epsilon(t),$$

where the errors ϵ are independent and satisfy $E(X^4(t)) < \infty$. Let \hat{f}_p be the autoregressive spectral estimator of the spectral density f based on a realization of length n . Then

a) If p is chosen so that

$$i) p \rightarrow \infty, \quad ii) \frac{p^3}{n} \rightarrow 0, \quad \text{and} \quad iii) \sqrt{n} \sum_{j=p+1}^{\infty} |\alpha_j| \rightarrow 0,$$

as $n \rightarrow \infty$, then for any k fixed frequencies $0 < \omega_1 < \dots < \omega_k < 0.5$, the joint asymptotic distribution of

$$\begin{aligned} &\sqrt{n/p}(\hat{f}_p(0) - f(0)), \sqrt{n/p}(\hat{f}_p(\omega_1) - f(\omega_1)), \dots, \\ &\sqrt{n/p}(\hat{f}_p(\omega_k) - f(\omega_k)), \sqrt{n/p}(\hat{f}_p(0.5) - f(0.5)), \end{aligned}$$

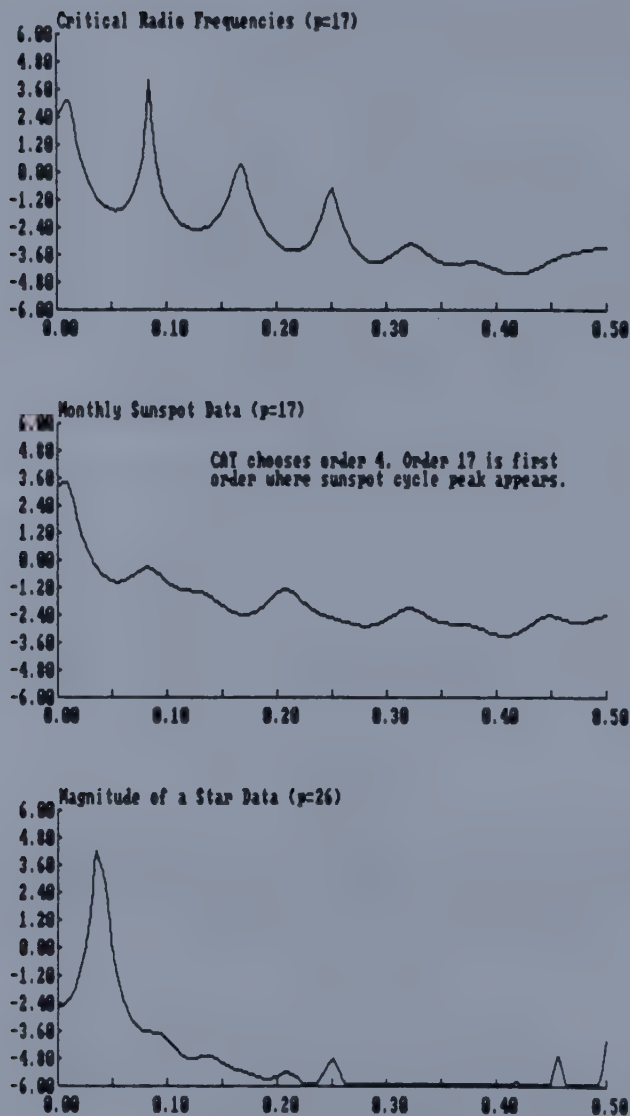


Figure 3.13. Autoregressive Spectral Estimators for the Three Series Analyzed in Figure 3.8.

is that of independent, zero mean, normal random variables having variances

$$4f^2(0), 2f^2(\omega_1), \dots, 2f^2(\omega_k), 4f^2(.5).$$

b) If X is a finite order autoregressive process, then letting p be any function g of n that satisfies $p \rightarrow \infty$ and $p^3/n \rightarrow 0$ as $n \rightarrow \infty$ satisfies the requirements of part (a), and the variance of $\hat{f}_p(\omega)$ goes to zero at the rate $g(n)/n$ as $n \rightarrow \infty$.

c) If X is an invertible ARMA process, then $p = \log n$ satisfies the requirements of part (a), and the variance of $\hat{f}_p(\omega)$ goes to zero at the rate of $\log n/n$ as $n \rightarrow \infty$.

Implications: This theorem shows that if we are estimating the spectral density of a process that can be written as an infinite order AR, then the autoregressive spectral estimator is consistent and asymptotically normal as long as the order that we use is chosen in the correct way. Requirement (iii) for the choice of p shows that the choice depends on how rapidly the coefficients of the $AR(\infty)$ go to zero. For a finite order AR process, these coefficients become zero past the order of the process, and we can choose p to go to infinity as slowly as we want. For an invertible MA or ARMA process, we could show that the coefficients of the $AR(\infty)$ representation go to zero at a rate bounded by ρ^k for some constant ρ and thus choosing $p = \log n$ is sufficient (see Problem T3.15).

The other important part of this theorem is the fact that the variance of the estimator goes to zero at the rate of p/n , so that if p can be chosen as a function of n that goes to ∞ very slowly (such as in parts (b) and (c)), then the variance goes to zero very rapidly. Again, this choice of p depends upon the speed at which the $AR(\infty)$ coefficients go to zero. Contrast this with the window case, where the rate of decrease of the variance is bounded by the characteristic exponent q of the window being used. This difference between the window estimator and the AR spectral estimator is often phrased that the window estimator is "saturated" (the decrease in variance is bounded by the nature of the window and not by the nature of the process), while the AR spectral estimator is not saturated. See Davis (1980) for more discussion of saturation.

As an example of this difference, suppose we are estimating the spectral density of an invertible ARMA process both by a window estimator using one of the lag window generators having $q = 2$, and by the AR spectral estimator using $p = \log n$. Then the two rates of decrease of variance are $n^{-4/5} = n^{1/5}/n$ and $\log n/n$, and the latter goes to zero faster than the former (see Problem C3.16).

Optimality of AIC and CAT for AR Spectral Estimation

We saw in Theorem 3.5.1 that the AIC (and thus the CAT criterion) tends to overestimate the order of a finite order autoregressive process. In the context of autoregressive spectral estimation, it turns out that the resulting spectral estimator is optimal in the following sense. Suppose for a given sample size n , we allow the order to be chosen from the range 1 through K_n where K_n is chosen as a function of n so that $K_n \rightarrow \infty$ and $K_n/\sqrt{n} \rightarrow 0$ as $n \rightarrow \infty$. We can measure how well the autoregressive spectral estimator estimates a true spectral density f by its integrated relative squared error

$$\text{IRSE}(\hat{f}_{\hat{p}}) = \int_0^1 \left(\frac{\hat{f}_{\hat{p}}(\omega) - f(\omega)}{f(\omega)} \right)^2 d\omega.$$

Now let $N = n - K_n$ and $C_n(k) = \sigma_k^2 - \sigma^2 + k\sigma^2/N$, where σ_k^2 and σ^2 are the memory k and infinite memory prediction error variances for the process whose spectral density is being estimated. Then for a nondegenerate (that is, not expressible as a finite order process) Gaussian infinite order AR process, Shibata (1981) has shown that the AIC is as good as any other possible order-determining criterion in terms of having a standardized form of the IRSE attain an asymptotic lower bound in probability. Since AIC and CAT are asymptotically equivalent if we let $\text{CAT}(0) = \hat{R}(0)$, we have that CAT shares this optimality.

The next theorem (due to Newton and Pagano (1984)) gives confidence bands for AR(p) spectra.

Theorem 3.8.4

CONFIDENCE BANDS FOR AR SPECTRA

If X is a Gaussian AR(p) process, then asymptotically the probability is $1 - \alpha$ that the true spectral density f lies entirely within the Scheffe-type bands (see part (b) of Theorem A.5.2)

$$\frac{1}{\hat{h}_p(\omega) + s(\omega)} \leq f(\omega) \leq \frac{1}{\hat{h}_p(\omega) - s(\omega)}, \quad \omega \in [0, 1],$$

where

$$\hat{h}_p(\omega) = \frac{1}{\hat{f}_p(\omega)} = \hat{\gamma}(v) + 2 \sum_{v=1}^p \hat{\gamma}(v) \cos 2\pi v\omega$$

$$\hat{\gamma}(v) = \frac{1}{\hat{\sigma}_p^2} \sum_{j=0}^{p-v} \hat{\alpha}_p(j) \hat{\alpha}_p(j+v), \quad v = 0, \dots, p$$

$$s^2(\omega) = \frac{\chi_{\alpha, p+1}^2}{n} \mathbf{x}^T(\omega) \hat{\mathbf{D}} \mathbf{x}(\omega)$$

and

$$\mathbf{x}^T(\omega) = (1, 2 \cos 2\pi\omega, \dots, 2 \cos 2\pi p\omega)$$

$$\mathbf{D} = \mathbf{B}\mathbf{C}\mathbf{B}^T$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{S}(\boldsymbol{\alpha}) & \mathbf{0}_p \\ \mathbf{0}_p^T & 2/\sigma^4 \end{bmatrix}$$

$$B_{j,k} = \begin{cases} \sigma^2 \gamma(j-1), & k = p+1, \quad j = 1, \dots, p+1 \\ \sigma^{-2} (\alpha_{k+j-1} + \alpha_{k-j+1}), & k = 1, \dots, p, \quad j = 1, \dots, p+1, \end{cases}$$

$\mathbf{S}(\boldsymbol{\alpha})$ is the Schur matrix corresponding to $\boldsymbol{\alpha}$, and in forming $\hat{\mathbf{D}}$ estimators of $\boldsymbol{\alpha}$ and σ^2 are substituted for the true values. For any frequency ω where $\hat{h}(\omega) - s(\omega)$ is negative, the upper limit is taken to be ∞ .

The ARSPCB Command

For an AR(p) process whose order, coefficients, and error variance are entered in the integer **p**, the array **alpha** of length **p**, and the real scalar **rvar**, and a confidence level $1-\alpha$ that is entered in the real scalar **conf**, the command

ARSPCB(alpha,p,rvar,n,Q,conf,fl,fu)

will return the lower and upper confidence bands in the arrays **fl** and **fu** at the $q = [Q/2] + 1$ frequencies of the form $(j-1)/Q$. The input integer **n** is the sample size to be used in the calculations.

To illustrate the ideas of this section, consider again the lynx data. In Figure 3.14 we give a plot of the AR(11) autoregressive spectral estimator for the log (base 10) of the data. We also superimpose the upper and lower 90% confidence bands (the macro that was used to produce this figure is given in Example 3.18). Notice that the upper limit for low frequencies has to be taken to be ∞ . We use this as an indication that the data may contain a deterministic component. See Section 3.9 for more discussion of this issue.

3.8.2. MA and ARMA Spectral Estimation

The next natural step in spectral estimation is to use MA and ARMA models to obtain estimators. We note that MA spectral estimation (see Brockwell and Davis (1985)) is very similar to nonparametric spectral estimation in that it results in an estimator that is a finite-degree trigonometric polynomial. Thus it suffers from the same difficulty in estimating spectra that have sharp peaks. It does have the advantage of having the AIC available to aid in choosing the degree of the polynomial.

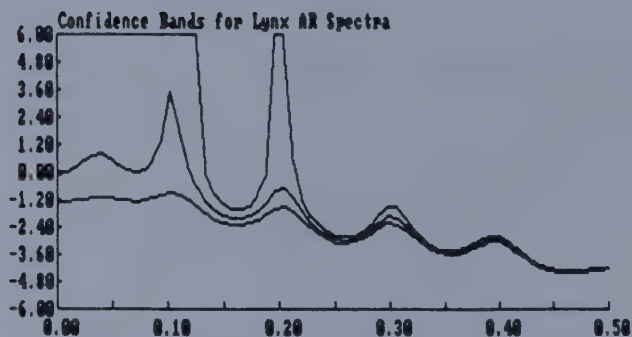


Figure 3.14. AR(11) Spectral Estimator and 90% Confidence Bands for the Log of the Lynx Data.

The use of ARMA models to estimate spectra does not have the weakness that nonparametric and moving average methods have and much attention has been paid recently to ARMA spectral estimation (see Gray and Woodward (1986) for example). However, unless the true spectral density has both sharp peaks and troughs, autoregressive spectral estimation, with its computational simplicity, should be adequate.

3.9. Methods for Determining Periodicities

Several of the most famous time series that have been studied in the past have appeared to contain cyclical components. For example, in Figure 3.15, we display two data sets: (1) the annual number of Canadian lynx trapped on the Mackenzie River for the years 1821–1934, and (2) the annual average value of the daily index of the number of sunspots (using the scale devised by Professor Rudolf Wolf in 1849) for the years 1755–1964. These data sets have been extensively studied over the past several years (see for example Part 4 of the 1977 volume of the *Journal of the Royal Statistical Society, Series A*). The basic property of these data is that there appear to be cyclic patterns but that these patterns are not perfectly cyclic. The “sunspot cycle” is a well known phenomenon and has an important effect on radio communications. The cycle in the lynx data is usually explained as being due to a predator-prey relationship between the Canadian lynx and the snowshoe hare, its most important source of food.

Time series such as the lynx and sunspot data are traditionally analyzed according to one of three models:

1. As the sum of a deterministic sinusoid plus an error that is a covariance

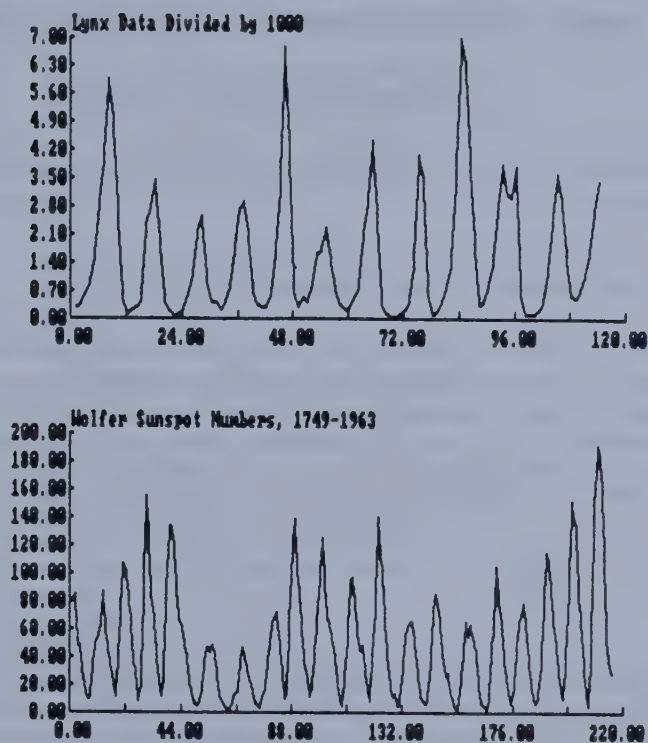


Figure 3.15. The Lynx and Sunspot Data.

stationary time series, that is,

$$X(t) = \mu + a \cos\left(\frac{2\pi t}{p}\right) + b \sin\left(\frac{2\pi t}{p}\right) + \epsilon(t),$$

where the error series ϵ is covariance stationary, and p is the period of the sinusoid.

2. As the values of a stochastic process referred to by names such as an 'outburst' model (see Morris (1977)) or a filtered Poisson process (see Parzen (1962b)). The basic idea of this type of model is that a physical process builds up and then has a large surge or outburst, followed by a decay back to some baseline value. We will not consider this model further.
3. As the values of a covariance stationary time series of near cyclic type.

For example, in Problem T2.7 we discussed the fact that the AR(2) model

$$X(t) + \alpha_1 X(t-1) + \alpha_2 X(t-2) = \epsilon(t)$$

will appear roughly cyclic with period p if $\alpha_1 = -2\cos(2\pi/p)$ and α_2 is close to 1.

3.9.1. Deterministic Sinusoid Plus Error

If we believe that a time series is actually a deterministic sinusoid observed with additive error and we know the period of the sinusoid, then we can use regression analysis to estimate the coefficients of the sinusoid and test whether the amplitude is in fact zero. If the errors are uncorrelated, then ordinary least squares can be used, while if the errors are correlated, we can use the procedure described in Section 3.7.1 for doing regression with autoregressive errors. In fact, in Section 3.7.1 we used the lynx data to illustrate the procedure.

Estimating a Period

If one has data $X(1), \dots, X(n)$ and suspects that they contain a deterministic sinusoid of some unknown frequency plus noise, then this problem is often referred to as a search for hidden periodicities. It seems natural to inspect the periodogram of the data and test whether the largest value of the periodogram is significantly different from zero. If not, then we would conclude that there are no deterministic sinusoids of any of the natural frequencies in the data. If the largest value is significantly different from zero, then we must ask whether it could happen that there is a sinusoid but it is actually at a different frequency. To do this test, we can use the results of the next theorem. In this theorem we assume that n is odd. The results are approximately correct if n is even.

Theorem 3.9.1	FISHER'S EXACT TEST
----------------------	----------------------------

Let $\hat{f}(\omega_j)$ be the periodogram of a realization of length n (n odd) from the process

$$X(t) = a \cos \frac{2\pi(t-1)}{p} + b \sin \frac{2\pi(t-1)}{p} + \epsilon(t),$$

where the period p is a factor of n , and ϵ is a Gaussian white noise series. Let $m = \lfloor n/2 \rfloor$. Then

a) If $a = b = 0$, the exact distribution of

$$g = \frac{\max_{j=2, [n/2]} (\hat{f}(\omega_j))}{\sum_{j=2}^{[n/2]} \hat{f}(\omega_j)}$$

is given by

$$\Pr(g > z) = \sum_{j=1}^K (-1)^{j-1} \binom{m}{j} (1-jz)^{m-1}, \quad z > 0,$$

where $K = [1/z]$.

b) The chance that the test given in part (a) will choose a frequency other than $1/p$ is less than the level of significance α that is used in the test.

The test in part (a) is called Fisher's (1929) exact test, while part (b) is due to Hartley (1949). In Example 3.17 we give a macro for calculating the p -value for Fisher's test. For the log of the lynx data, we find that $g = 0.5967$ which has a p -value less than 10^{-4} , and the maximum is at period $114/12 = 9.5$. If one suspects the existence of r sinusoids, then the above test can be modified (see Problem T3.16).

A crucial assumption in the Fisher test is that the errors are white noise. If they are not, that is, the spectral density of the error time series is not a constant, then a large value of the periodogram at a particular frequency could be due to either (1) the existence of a sinusoid having that frequency, (2) the spectral density being large at that frequency, or (3) a combination of the two. This is referred to as the mixed spectrum case (see Section 6.3 of Priestley (1981)), and unless one has some prior knowledge of the nature of f or the sinusoids involved, separating the two parts of the model is very difficult.

3.9.2. Estimating Peak Frequencies in AR Spectra

For a process of near cyclic type, the data appear to be cyclic except that the lengths of cycles vary from one cycle to the next. Thus such a model is often referred to as a disturbed periodicity model, and the analog of determining the period of a deterministic sinusoid is to determine the frequency ω^* where the spectral density of the process has a peak. Searching for peak frequencies is not difficult if the process is a finite order MA or AR process as then f (or its reciprocal in the AR case) is a finite-degree trigonometric polynomial and finding the critical values (maxima and minima) of such a polynomial is not difficult. Thus suppose that we have an $\text{AR}(p)$ process. This means that

$$h(\omega) = \frac{\sigma^2}{f(\omega)} = \left| \sum_{j=0}^p \alpha_j e^{2\pi i j \omega} \right|^2,$$

which is the spectral density function of an $MA(p)$ process having coefficients α and error variance 1. Note that the critical values for h are the same as those of f since h is the reciprocal of f . Thus peaks in f correspond to troughs in h and troughs in f correspond to peaks in h . Because h is an MA spectral density, we have

$$h(\omega) = \gamma(0) + 2 \sum_{v=1}^p \gamma(v) \cos 2\pi v \omega,$$

where

$$\gamma(v) = \sum_{j=0}^{p-v} \alpha_j \alpha_{j+v}, \quad v = 0, \dots, p.$$

Finding where a relative maximum of f occurs corresponds to finding the location of the corresponding relative minimum of h . Thus we need to find the frequency ω^* where the derivative h' is zero. Given an initial guess ω_0 for such a zero, we can apply Newton's root finding method to find ω^* as the limit of the sequence $\omega_0, \omega_1, \dots$, where

$$\omega_j = \omega_{j-1} - \frac{h'(\omega_{j-1})}{h''(\omega_{j-1})}.$$

The first and second derivatives of h are given by

$$h'(\omega) = -4\pi \sum_{v=1}^p v \gamma(v) \sin 2\pi v \omega \quad \text{and} \quad h''(\omega) = -8\pi^2 \sum_{v=1}^p v^2 \gamma(v) \cos 2\pi v \omega.$$

Given a realization of length n from an $AR(p)$ process, we can estimate the peak frequency ω^* by using the above procedure with estimates of the α 's replacing the true values. We will denote this estimator by $\hat{\omega}$. If the order p is unknown, then we can estimate it and the coefficients of the estimated order process and again use the process described above to estimate the peak frequency. This estimator is denoted by $\hat{\omega}_{\hat{p}}$. The properties of these estimators are given in the next theorem.

Theorem 3.9.2	ESTIMATING A PEAK FREQUENCY
----------------------	------------------------------------

Let $\hat{\omega}$ and $\hat{\omega}_{\hat{p}}$ be the autoregressive spectral estimates described above of a peak frequency ω^* based on a realization of size n from a Gaussian $AR(p, \alpha, \sigma^2)$ process. Then

a) $\sqrt{n}(\hat{\omega} - \omega^*) \xrightarrow{L} N(0, \sigma^2(\omega^*))$, where

$$\sigma^2(\omega^*) = \frac{\mathbf{b}^T(\omega^*) \mathbf{C}(\alpha) \mathbf{S}(\alpha) \mathbf{C}^T(\alpha) \mathbf{b}(\omega^*)}{(h''(\omega^*)/2\pi)^2},$$

where $S(\alpha)$ is the Schur matrix corresponding to α ,

$$\mathbf{b}^T(\omega) = (\sin 2\pi\omega, 2 \sin 4\pi\omega, \dots, p \sin 2\pi p\omega),$$

$h(\omega) = \sigma^2/f(\omega)$, and $C(\alpha)$ is the $p \times p$ matrix having (j, k) th element

$$C_{j,k}(\alpha) = \alpha_{j+k} + \alpha_{k-j},$$

with $\alpha_v = 0$ if $v > p$ or $v < 0$.

b) If a consistent order-determining criterion is used for finding \hat{p} , then the results of part (a) continue to hold for $\hat{\omega}_{\hat{p}}$.

c) If an order-determining criterion that is guaranteed asymptotically to not underestimate the order is used to find \hat{p} , then $\hat{\omega}_{\hat{p}}$ is a consistent estimator of ω^* .

Part (a) of this theorem is given in Newton and Pagano (1983a), while parts (b) and (c) are given in Ensor and Newton (1987). From these results, we can find a large sample confidence interval for ω^* as

$$\hat{\omega} \pm Z_{\alpha/2} \sigma(\hat{\omega}),$$

while the lower and upper limits in a confidence interval for the reciprocal of ω^* , that is, for the period of the peak, are given by the reciprocals of the upper and lower limits for the frequency. Notice the important role played by the second derivative of h in the asymptotic variance of $\hat{\omega}$. For a sharp peak, h'' will be large and thus the confidence interval will be narrow. For a broad peak, the interval will be wide. We should also note the similarity of this problem to the problem of estimating the mode of a continuous probability distribution (see Parzen (1962a)).

A crucial part of the method of this section is the fact that we have been finding a zero of a finite-degree polynomial. Thus if f is of the form of an ARMA spectral density, that is, as the ratio of two finite-degree polynomials, we cannot apply the above procedure. However, as long as f can be expressed as the spectral density of either an AR(∞) or MA(∞) process, we should be able to apply the above procedure and obtain asymptotically good properties.

The ARSPPEAK Command

Given the parameters p , α , and σ^2 of an AR process in the integer \mathbf{p} , the array \mathbf{alpha} , and the real scalar \mathbf{rvar} , the command

```
ARSPPEAK(alpha,p,rvar,n,ier,freq,se[,start])
```

will attempt to return in the real scalar \mathbf{freq} a critical value of the corresponding spectral density, and in the real scalar \mathbf{se} , the value of $\sigma(\omega)$ corresponding

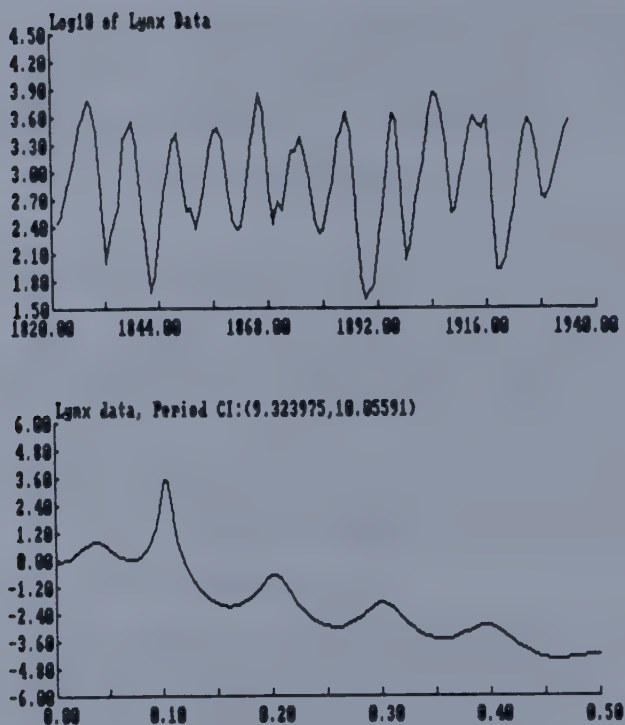


Figure 3.16. Log (base 10) of the Lynx Data and the Estimated AR(11) Spectral Density.

to the frequency ω and the sample size that was entered in the integer n . Thus if ARSPPEAK is not being used for estimation, then n should be given the value 1. The optional argument `start` is a real scalar input argument specifying a starting value for the root-finding procedure. If no starting value is included in the call to ARSPPEAK, then the location of the largest relative maximum of the spectral density evaluated at 256 frequencies between 0 and 1 will be used as the starting value. The output integer `ier` is an error indicator. A value of 0 means that ARSPPEAK was successful in finding a peak. Values of 1, 2, 3, and 4 indicate respectively that no relative maximum was found, that a zero second derivative was encountered, that the iterations converged to frequency 0 or .5, and that the iterations did not converge. All of these positive values of `ier` result in no other output being formed. If f has more than one peak, then ARSPPEAK can be called more than once, with different starting values being used each time.

To illustrate the use of ARSPPEAK, we consider the lynx data again. While there is no clear agreement among analysts what AR process best models the log of the lynx data, we consider the result of using the order 11 process determined in Section 3.4. In Figure 3.16 we give plots of the log (base 10) of the data and the estimated AR(11) spectral density. The confidence interval for the period corresponding to the largest peak in the spectral density is (9.32, 10.05) years. The macro that produced these plots is given in Example 3.19.

3.10. Examples and Problems

Example 3.1

CONFIDENCE INTERVALS FOR THE MEAN

The macro CIS.MAC given below was used to find the graphs in Figure 3.2.

```

1 ;;
2 ;;   CIS.MAC: macro to generate nsamps realizations of length
3 ;;           n from an AR(1) process having coefficient alpha,
4 ;;           and find the 95 % confidence interval for mu=0 for
5 ;;           each one, both ignoring and not ignoring the effect
6 ;;           of the autocorrelation.
7 ;;
8 ;;   INPUT: nsamps, n, alpha (real scalar)
9 ;;           seed (seed for random number generator)
10 ;;
11 PAUSE
12 ;start
13 ul=LINE(nsamps,0,0)           ;Reserve arrays to receive quantities
14 ll=LINE(nsamps,0,0)           ;to be used to find limits.
15 xbar=LINE(nsamps,0,0)         ;We do it this way so that the macro
16 rho1=LINE(nsamps,0,0)         ;will go faster.
17 r01=LINE(nsamps,0,0)
18 alph=<alpha>
19 ns=1
20 x=wn(seed,10)                ;warm up generator
21 PROMPTOFF                    ;turn off prompt
22 ;
23 ;s1
24 ;
25 LIST(ns)                     ;tell the user which sample is being done
26 x=ARDT(alph,1,1,0,n,ier,r0)   ;generate realization
27 x=SUBMWS(x,n,1,xbar[ns])      ;find mean
28 rho= CORR(x,n,1,0,1,r01[ns],per) ;find rho(1)
29 rho1[ns]=rho[1]
30 IF(ns.eq.nsamps,s2)
31 ns=ns+1
32 GOTO(s1)
33 ;
34 ;s2

```

```

35 ; Form the plots:
36 ;
37 lim=1.96*(r01/n)^.5
38 low1=xbar-lim ;low1 and up1 are ignoring autocorrelation
39 up1=xbar+lim
40 LABEL(low1)='Ignoring Correlation'
41 LABEL(up1)=' '
42 ;
43 ; Form the plot incorporating autocorrelation:
44 ;
45 ul=({1+rho1}/{1-rho1})^.5
46 lim=lim*ul ;the limits now are just SQRT((1+rho)/(1-rho))
47 low2=xbar-lim ;low2 and up2 are not ignoring autocorrelation
48 up2=xbar+lim
49 ll=<low1,low2> ;get max's and min's so plots on same scale
50 n2=2*nsamps
51 MAXMIN(ll,n2,lmax,im,lmin,im)
52 ll=<up1,up2>
53 MAXMIN(ll,n2,umax,im,umin,im)
54 LABEL(low2)='Incorporating Correlation'
55 LABEL(up2)=' '
56 PLOT2(low1,up1,nsamps,nsamps,3,3,0,n,lmin,umax)
57 PLOT2(low2,up2,nsamps,nsamps,3,3,0,n,lmin,umax)

```

Example 3.2 MSE FOR $\hat{\rho}$ AND $\tilde{\rho}$

The macro MSERHO.MAC given below can be used to compare the performance of the biased and unbiased estimates of the autocorrelations of a user-specified AR process. A series of realizations are generated and the mean square error of both the biased and unbiased estimates of the autocorrelation function of the process is determined. The macro also keeps track of which estimator is closest to the true values.

```

1 ;;
2 ;; MSERHO.MAC: macro to generate a series of nsamps realizations
3 ;; of length n from an AR(p) process having coefficients
4 ;; alpha and error variance 1. For each realization,
5 ;; mu=10 is added, then xbar is subtracted, and the
6 ;; biased and unbiased autocorrelations calculated.
7 ;;
8 ;; The average squared difference from the true
9 ;; correlations is calculated as well as the number
10 ;; of times for each lag that the biased estimate
11 ;; is closer.
12 ;;
13 ;; INPUT: nsamps, n, p, alpha, seed, N (number of correlations)
14 ;;
15 PAUSE
16 ;start
17 PROMPTOFF

```

```

18  smp#=1
19  x=WM(seed,10)                ;warm up random number generator
20  rhot=ARCORR(alpha,p,1,M,r0,ier) ;find true correlations
21  nclose=LINE(M,0,0)            ;*****
22  mseb=LINE(M,0,0)              ;
23  mseu=LINE(M,0,0)              ;Reserve space and find factor
24  fac=LINE(M,0,1)              ;
25  fac=n/{n-fac}                ;*****
26  ;
27  ;startloop
28  ;
29  LIST(smp#)                    ;tell user which sample we're doing
30  x=ARDT(alpha,p,1,0,n,ier,r0) ;generate sample
31  x=x*10.                      ;add 10
32  rhob=CORR(x,n,M,0,1,r0,per)   ;find biased correlations
33  rhou=rhob*fac                 ;multiply to get unbiased
34  db=rhob-rhot                  ;difference for biased
35  du=rhou-rhot                  ;difference for unbiased
36  adb=ABS(db,M)
37  adu=ABS(du,M)
38  aa=adb-adu                    ;negative element means biased is closer
39  aa=REPLACE(aa,M,5,0,1,0)     ;this replaces negative by 1, positive by 0
40  nclose=nclose+aa             ;update number closer
41  mseb=mseb+adb^2              ;update sum of squared error
42  mseu=mseu+adu^2
43  IF(smp#.eq.nsamps,endoloop)   ;finished?
44  smp#=smp#+1                  ;no
45  GOTO(startloop)
46  ;
47  ;endloop                      ;yes
48  ;
49  mseb=mseb/nsamps              ;divide and display results
50  mseu=mseu/nsamps
51  nclose=nclose/nsamps
52  ;list
53  LABEL(alpha)='AR Coefficients'
54  LIST(alpha)
55  LABEL(rhot)='True Correlations'
56  LIST(rhot)
57  LABEL(nclose)='Proportion of Times Biased Closer'
58  LIST(nclose)
59  LABEL(mseb)='MSE for Biased Correlations'
60  LABEL(mseu)='MSE for Unbiased Correlations'
61  LIST(mseb)
62  LIST(mseu)
63  PROMPT

```

We used this macro for the AR(2) process having coefficients 0.3 and 0.9 and for the AR(1) process having coefficient -0.5 and obtained the results given in Table 3.10. Note that the biased estimate is closer to the true correlation and has a smaller average squared error whenever the true autocorrelation is large,

Table 3.10. Example Output from the MSERHO Macro

AR Coefficients					
1	.300000		.900000		
True Correlations					
1	-.157895	-.852632	.397895	.648000	-.552505
6	-.417448	.622489	.188957	-.616927	.015017
Proportion of Times Biased Closer					
1	.570000	.790000	.580000	.710000	.650000
6	.680000	.760000	.480000	.790000	.060000
MSE for Biased Correlations					
1	.000526	.002977	.004097	.009134	.011293
6	.013806	.021352	.016190	.032362	.019104
MSE for Unbiased Correlations					
1	.000523	.004156	.004374	.011518	.013104
6	.015141	.026104	.014926	.039445	.015455
AR Coefficients					
1	-.500000				
True Correlations					
1	.500000	.250000	.125000	.062500	.031250
6	.015625	.007813	.003906	.001953	.000977
Proportion of Times Biased Closer					
1	.600000	.560000	.390000	.190000	.100000
6	.070000	.030000	.010000	.000000	.000000
MSE for Biased Correlations					
1	.009776	.015815	.013685	.017076	.016424
6	.018038	.013897	.011512	.009919	.013561
MSE for Unbiased Correlations					
1	.009832	.015528	.013081	.015893	.014936
6	.015994	.012045	.009758	.008220	.010989

even if they are for a large lag. This was somewhat unexpected since the bias in the biased estimator should get larger with the lag.

Example 3.3

STANDARD ERROR OF AUTOCORRELATIONS

Suppose that there is a lag q for which the true autocorrelation function $\rho(v)$ of a time series X is essentially zero for $|v| > q$. Then Bartlett's formula for the variance of $\hat{\rho}(v)$ (see part (c) of Theorem 3.1.3) can be approximated by

$$\text{Var}(\hat{\rho}(v)) \doteq \frac{1}{n} \left[1 + 2 \sum_{k=1}^q \rho^2(k) \right],$$

which is independent of v . In practice, one replaces the true autocorrelations in this expression by sample autocorrelations and then superimposes on the correlogram lines at plus and minus the square root of the resulting estimated variance. The macro given below will carry out this procedure for a user-specified value of q . To illustrate its use, consider Figure 3.17 which is the correlogram with the superimposed standard error lines for the rainfall data

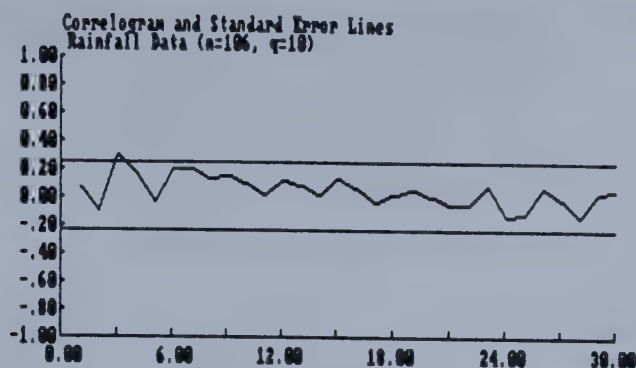


Figure 3.17 Correlogram and Standard Error Lines for the Rainfall Data.

set (Series IV in Chapter 1). The STDRHO macro is appropriate for this data set as its correlogram is small for all lags.

```

1 ;;
2 ;;   STDRHO.MAC: macro to plot the correlogram of a series
3 ;;               of length M and superimpose the standard
4 ;;               error lines using 'truncation point' q (le M).
5 ;;
6 ;;   INPUT: x, n (data and sample size)
7 ;;          M, q
8 ;;
9 PAUSE
10 ;start
11 rho=CORR(x,n,M,0,1,r0,per) ;find correlations
12 ss=DOT(rho,rho,q)
13 std2=2*{(1+2*ss)/n}^.5      ;find two times standard error
14 mstd2=-std2
15 xx=LINE(M,0,1)              ;construct input to PLOTK
16 xx=<xx,0,M,0,M>
17 yy=<rho,std2,std2,mstd2,mstd2>
18 nn=<M,2,2>
19 type=<2,2,2>
20 LABEL(yy)='Correlogram and Standard Error Lines'
21 LABEL(xx)='n=#n#, q=#q#'
22 PLOTK(xx,yy,nn,3,type,0,M,-1,1)

```

Example 3.4 INCONSISTENCY OF THE PERIODOGRAM

In this simple macro, we generate realizations of length 200 through 1000 in steps of 200 from a Gaussian white noise process having variance 1. For

each sample we graph the log of the periodogram. In each case we use 1 as the value to standardize the periodogram so that they are all on the same scale.

```

1 ;;
2 ;;   INCONS.MAC: macro to illustrate the inconsistency of the
3 ;;               periodogram.
4 ;;
5 ;;   INPUT: seed (seed for random number generator)
6 ;;
7 ;;
8 PAUSE
9 ;start
10 PLOTOM
11 n=200
12 x=WN(seed,10)    ;warm up generator
13 ;
14 ;startloop
15 ;
16 x=WN(0,n)
17 rho=CORR(x,n,0,n,1,r0,per)
18 LABEL(per)='White Noise Periodogram, n=#n#'
19 CLS
20 PLOTSP(per,n,1) ;note that we're standardizing by 1 each time
21 IF(n.eq.1000,endoloop)
22 n=n+200
23 GOTO(startloop)
24 ;
25 ;endoloop
26 ;
27 PLOTOFF

```

Example 3.5

THE FÉJER KERNEL

The macro FEJER.MAC calculates the Féjer kernel

$$F_n(\omega) = \frac{1}{n} \left(\frac{\sin n\pi\omega}{\sin \pi\omega} \right)^2$$

for the Q equally spaced frequencies between f_1 and f_2 inclusive. To construct Figure 3.2 we used the macro twice for each n : once to get the graph for $[-.1, .1]$ and once for $[.1, .5]$.


```

1 ;;
2 ;; FEJER.MAC: macro to calculate the Fejer kernel for
3 ;; user-specified n for Q frequencies between
4 ;; frequencies f1 and f2 (inclusive). The
5 ;; kernel is then plotted with the vertical
6 ;; scale being between 0 and ymax.
7 ;;
8 ;; INPUT: n, Q (number of frequencies between f1 and f2)
9 ;; f1, f2, ymax
10 ;;
11 ;;
12 PAUSE
13 ;start
14 freq=LINE(Q,f1,f2,a) ;now we have frequencies
15 ff=pi*freq
16 ff=REPLACE(ff,Q,4,0,0,.0001) ;make sure frequency 0 not included
17 den=SIN(ff,Q) ;denominator
18 ff=n*ff
19 num=SIN(ff,Q) ;numerator
20 fejq={num/den}^2/n
21 LABEL(fejq)='Fejer Kernel For n=#n#'
22 LABEL(freq)=' '
23 PLOT(freq,fejq,Q,f1,f2,0,ymax)

```

Example 3.6 CDF OF THE BARTLETT STATISTIC

The macro BARTCDF.MAC macro given below was used to find the values used in Figure 3.4 and Table 3.1.

```

1 ;;
2 ;; BARTCDF.MAC: Macro to find the cdf Pr(B.le.b) of the
3 ;; Bartlett statistic for b=.40,.41,...,2.00
4 ;;
5 ;; INPUT: None
6 ;;
7 PAUSE
8 PROMPTOFF ;turn prompt off
9 cdf=LINE(161,0,0) ;reserve space for cdf
10 b=LINE(161,.4,2,1) ;define the 161 values of b in [.4,2]
11 i=1
12 ;
13 ;start
14 ;
15 pval=BARTTEST(b[i])
16 cdf[i]=1.-pval
17 LIST(b[i],cdf[i])
18 if(i.eq.161,plot)
19 i=i+1
20 GOTO(start)
21 ;
22 ;plot

```

```

23 ;
24 LABEL(cdf)='CDF of the Bartlett Statistic'
25 LABEL(b)=' '
26 PLOT(b,cdf,161,.4,2.0,0,1)
27 bb=<b,cdf> ;this is how we mix two arrays
28 bb=TRANS(bb,161,2)
29 LABEL(bb)='CDF of Bartlett Statistic'
30 LIST(bb,322,8,8f6.3) ;we could use RECORD to send this to
31 PROMPT ;a file.

```

Example 3.7 THE CHOICE OF m IN THE Q TEST

The two macros given below were used to find the results in Section 3.2.1 for studying the effect of choosing m . The macro QLEV.MAC plays the role of a main program, while QLEVEL.MAC is called repeatedly by QLEV.MAC.

```

1 ;;
2 ;;   QLEV.MAC: macro to repeatedly call QLEVEL.MAC
3 ;;               to form Table 3.2.
4 ;;
5 ;;   INPUT: seed1, nsamps
6 ;;           mm (array of length 3 with m's to use)
7 ;;           nn (array of length 3 with n's to use)
8 ;;           aa (array of length 3 with alpha's to use)
9 ;;
10 ;;
11 PAUSE
12 ;start
13 m1:=mm[1] ;note that we have to convert type
14 m2:=mm[2] ;from real to integer
15 m3:=mm[3]
16 seed=seed1
17 x=WN(seed,10) ;warm up random number generator
18 seed=0.
19 ;
20 ;
21 i=1 ;now we have a double loop over alpha and n
22 ;s1 ;we keep track of results using RECORD
23 j=1
24 ;s2
25 alpha=aa[i]
26 n:=nn[j]
27 MACRO(qlevel,start)
28 RECORD(qlevel.dat)
29 LIST(alpha,n,nrej1,nrej2,nrej3,ndiff)
30 RECORD(close)
31 IF(j.eq.3,e1)
32 j=j+1
33 GOTO(s2)
34 ;e1

```

```

35 IF(i.eq.3,e2)
36 i=i+1
37 GOTO(s1)
38 ;e2

1 ;;
2 ;; QLEVEL.MAC: Macro to study the effect of m on the significance
3 ;; level of the Q test.
4 ;;
5 ;; INPUT: nsamps,n (number of samples and sample size)
6 ;; seed (seed for WW)
7 ;; m1,m2,m3 (three different values of m (in increasing order))
8 ;; alpha (nominal significance level)
9 ;;
10 ;; OUTPUT: nrej1,nrej2,nrej3,ndiff (number of rejections for
11 ;; the different m's and the number of times the decisions
12 ;; are different)
13 ;;
14 PAUSE
15 ;start
16 pv1=LINE(nsamps,0,0) ;reserve space for p-values
17 pv2=LINE(nsamps,0,0)
18 pv3=LINE(nsamps,0,0)
19 x=WW(seed,10) ;warm up generator
20 ns=1 ;ns counts samples
21 PROMPTOFF
22 ;
23 ;startloop
24 ;
25 x=WW(0,n) ;generate sample
26 rho=CORR(x,n,m3,0,1,r0,per) ;find correlations
27 pv1[ns]=QTEST(rho,m1,0,0,n,Q) ;find p-values
28 pv2[ns]=QTEST(rho,m2,0,0,n,Q)
29 pv3[ns]=QTEST(rho,m3,0,0,n,Q)
30 LIST(ns) ;display sample number so that user
31 if(ns.eq.nsamps,endoloop) ;will know something is happening
32 ns=ns+1
33 GOTO(startloop)
34 ;
35 ;endoloop
36 ;
37 PROMPTON ;turn prompt back on
38 pv1=pv1-alpha ;subtract alpha from each vector
39 pv2=pv2-alpha
40 pv3=pv3-alpha
41 x1=EXTRACT(pv1,nsamps,0.0,lt,nrej1) ;this pulls off indices of
42 x2=EXTRACT(pv2,nsamps,0.0,lt,nrej2) ;significant p-values and
43 x3=EXTRACT(pv3,nsamps,0.0,lt,nrej3) ;counts how many there are
44 y1=LINE(nsamps,0,0) ;y1,y2,y3 will be arrays of 0's except
45 y2=LINE(nsamps,0,0) ;for places where there are significant
46 y3=LINE(nsamps,0,0) ;p-values
47 ABORTOFF ;make sure macro doesn't abort if no rejects

```

```

48 y1=REPLACE(y1,x1,1,nrej1) ;these put 1's in the right places
49 y2=REPLACE(y2,x2,1,nrej2)
50 y3=REPLACE(y3,x3,1,nrej3)
51 ABORTON                               ;now any error should cause macro to abort
52 y=y1+y2+y3                           ;an element of y will be 0 or 3 if no difference
53 x1=EXTRACT(y,nsamps,1.0,eq,nd1)      ;nd1=# of 0's
54 x2=EXTRACT(y,nsamps,2.0,eq,nd2)      ;nd2=# of 3's
55 ndiff=nd1+nd2

```

Example 3.8 AVERAGING THE PERIODOGRAM

The macro AVEPER.MAC was used to do the periodogram averaging procedure in Section 3.3.1.

```

1 ;;
2 ;;   AVEPER.MAC: macro to produce a smoothed periodogram estimate
3 ;;               of a spectral density. The ordinary average of
4 ;;               2m+1 values of the periodogram is used.
5 ;;
6 ;;   INPUT: per, m (periodogram and the number of frequencies on each
7 ;;               side of the target frequencies to be used)
8 ;;
9 PAUSE
10 ;start
11 LENGTH(per,q)
12 IF(m.eq.1)                               ;have to handle m=1 separately
13     ft=<per[2],per,per[q]>
14     GOTO(s1)
15 ENDDIF
16 mp1=m+1
17 pp=EXTRACT(per,2,mp1) ;put reversed values of 2nd thru (m+1)st
18 pp=REVERSE(pp,m)      ;at beginning
19 ft=<pp,per>
20 qmm=q-m
21 qmmp1=qmm+1
22 pp=EXTRACT(per,qmmp1,q) ;now reflect last m values of per
23 pp=REVERSE(pp,m)
24 ft=<ft,pp>
25 ;s1                                     ;*****
26 m2=2*m                                ;
27 qpm2=q+m2                             ; We use the FILT command to do the
28 m2p1=m2+1                             ; averaging.
29 w0=1.                                  ;
30 wt=LINE(m2,1,0)                       ;*****
31 ft=FILT(ft,wt,w0,qpm2,m2)
32 ft=ft/m2p1

```

Example 3.9	SPECTRAL WINDOWS
--------------------	-------------------------

This macro will calculate and display any of the eight spectral windows given in Table 3.3 and available in the **WINDOW** command. The window is found by using the **WINDOW** command with a correlation sequence of ones as input.

```

1 ;;
2 ;;   WINDOW.MAC: Macro to calculate and plot the spectral window
3 ;;           specified by the integer iopt (which corresponds
4 ;;           to the list in Table 3.3) for parameters n and M.
5 ;;
6 ;;           The window is calculated for Q frequencies between
7 ;;           0 and 1 and is plotted for frequencies between
8 ;;           f1 and f2. The vertical scale is from ymin to ymax.
9 ;;
10 ;;   INPUT: n, M, iopt, Q, f1, f2, ymin, ymax
11 ;;
12 PAUSE
13 ;start
14 q={Q/2}+1                ;Form frequencies
15 freq=LINE(q,0,.5,1)
16 rho=LINE(n,1,0)
17 r0=1.
18 w=WINDOW(rho,r0,M,Q,iopt,n) ;now have window at all frequencies
19 wi=EXTRACT(freq,q,f1,ge,ny) ;*****
20 w1=EXTRACT(w,wi,ny)        ;
21 freq1=EXTRACT(freq,wi,ny)   ; Now pull off values and frequencies
22 wi=EXTRACT(freq1,ny,f2,le,ny) ; between f1 and f2
23 w1=EXTRACT(w1,wi,ny)        ;
24 freq1=EXTRACT(freq1,wi,ny)  ;*****
25 LABEL(w1)='Window #iopt#, n=#n#, M=#M#'
26 LABEL(freq1)='      '
27 PLOT(freq1,w1,ny,f1,f2,ymin,ymax)

```

Example 3.10	WINDOW ESTIMATORS
---------------------	--------------------------

The macro **WINDSP.MAC** given below will find and plot the window spectral estimate (and confidence intervals) for a data set using user-specified scale parameter, window, and number of frequencies. The macro **WINDSP3.MAC** will form a graph of the Parzen window spectral estimator for a data set using three different truncation points. This macro was used to form Figure 3.8.

```

1 ;;
2 ;;   WINDSP.MAC: macro to find nonparametric spectral estimator
3 ;;               for a series x of length n.
4 ;;
5 ;;   INPUT: x, n
6 ;;           M (scale parameter), Q (number of frequencies)
7 ;;           ioptw (window to use)
8 ;;
9 PAUSE
10 ;start
11 mm=M                ;input depends on whether truncated form is used
12 IF(ioptw.le.5,s1)
13 mm=n-1
14 ;s1
15 rho=CORR(x,n,mm,0,1,r0,per)
16 spec=WINDOW(rho,r0,M,Q,ioptw,n,c)
17 spec1=spec/c
18 specu=spec*c
19 LABEL(spec)='Window Estimator and Conf Intervals'
20 LABEL(spec1)='M = #M#, Window Number #iopts#'
21 LABEL(specu)='<x>'
22 PLOTSP(spec1,Q,r0,spec,Q,r0,specu,Q,r0)

1 ;;
2 ;;   WINDSP3.MAC: macro to calculate Parzen window spectral estimate
3 ;;               for three truncation points.
4 ;;
5 ;;   INPUT: x, n (data and sample size)
6 ;;           M1, M2, M3 (truncation points, must be even and M3 biggest)
7 ;;           Q (number of frequencies)
8 ;;
9 PAUSE
10 ;start
11 rho=CORR(x,n,M3,0,1,r0,per)
12 f1=WINDOW(rho,r0,M1,Q,4)
13 f2=WINDOW(rho,r0,M2,Q,4)
14 f3=WINDOW(rho,r0,M3,Q,4)
15 LABEL(f2)='<x>'
16 LABEL(f1)='Parzen Window Spectral Estimate'
17 LABEL(f3)='M1 = #M1#, M2 = #M2#, M3 = #M3#'
18 PLOTSP(f3,Q,r0,f2,Q,r0,f1,Q,r0)

```

Example 3.11 ESTIMATING THE INNOVATION VARIANCE

Sometimes we would like to estimate the variance of the innovations process for a purely nondeterministic time series, that is, estimate

$$\sigma_{\infty}^2 = \exp\left(\int_0^1 \log f(\omega) d\omega\right)$$

(see part (d) of Theorem 2.4.1), given a realization $X(1), \dots, X(n)$. A natural way to estimate σ_∞^2 is by

$$\hat{\sigma}_\infty^2 = \exp\left(\frac{1}{n} \sum_{k=1}^n \log \hat{f}(\omega_k)\right),$$

where \hat{f} is the periodogram. Davis and Jones (1968) have shown that this estimator is in fact asymptotically biased, but that if one adds Euler's constant $\gamma = .57721$ to the exponent, then the resulting estimator is asymptotically unbiased. The following macro will find this unbiased estimator of the innovation variance. For more information about this problem, see Hannan and Nichols (1977) and Pukkila and Nyquist (1985).

```

1 ;;
2 ;;   INNOV.MAC: macro to estimate the innovation variance rvar
3 ;;           by the Davis-Jones method.
4 ;;
5 ;;   INPUT: x, n
6 ;;
7 ;;   OUTPUT: rvar, (Davis-Jones estimate)
8 ;;
9 PAUSE
10 ;start
11 x=SUBMWS(x,n,1,xbar)    ;find periodogram via FFT
12 FFT(x,n,n,1,zr,zi)
13 fhat={zr^2+zi^2}/n
14 fhat[1]=1.    ;this has the effect of leaving out frequency 0
15 lfhat=LOGE(fhat,n)
16 lfhat=SUBMWS(lfhat,n,1,rvar,0) ;this approximates integral
17 rvar=rvar+.57721          ;add bias correction
18 rvar=EXP(rvar)
19 LIST(rvar)

```

Example 3.12 YULE-WALKER AND BURG ESTIMATORS

As we pointed out in Section 3.4.4, the Yule-Walker estimation procedure can perform poorly for AR models that are ill-conditioned, that is, close to non-stationary. An extreme example of this is the AR(4) model having coefficients $-2.7607, 3.8106, -2.6535$, and $.9238$. In Figure 3.18 we give the log of the true spectral density of this process. Note that it contains a pair of sharp peaks that are quite close together. The fact that this model is ill-conditioned is emphasized by the fact that the zeros are given by:

Real Part of Polynomial Roots				
1	.649963	.649963	.786224	.786224
Imaginary Part of Polynomial Roots				
1	-.785937	.785937	.650041	-.650041

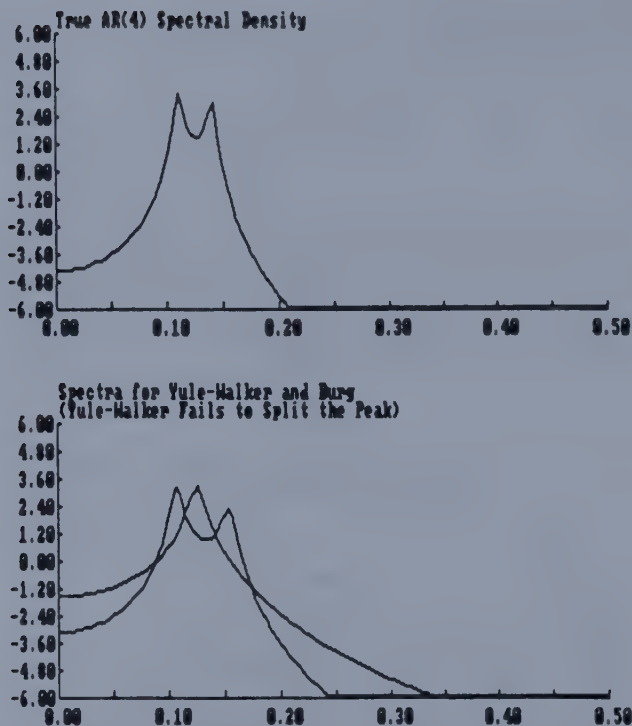


Figure 3.18. True AR(4) Spectral Density and the Burg and Yule-Walker Estimates for One Realization of Length 100.

Modulus of Roots

1	1.019877	1.019877	1.020148	1.020148
---	----------	----------	----------	----------

Thus the zeros are complex (which will make the data sinusoidal), close to the unit circle (and thus the process close to nonstationary), and also close together.

The following macro will generate a series of realizations from any user-specified AR process, and then use both Yule-Walker and Burg estimates of its coefficients to estimate the true spectral density.

```

1 ;;
2 ;;   BURG.W.MAC: macro to superimpose the Yule-Walker and
3 ;;           Burg estimates of the spectral density of
4 ;;           an AR(p) process.
5 ;;
6 ;;           nsamps realizations of length n are generated
7 ;;           from the process having coefficients alpha and
8 ;;           error variance 1.
9 ;;
10 ;;   INPUT: p, alpha, nsamps, n, seed
11 ;;
12 PAUSE
13 ;start
14 ns=1
15 x=WN(seed,10) ;warm up random number generator
16 ;
17 ;s1
18 ;
19 x=ARDT(alpha,p,1,0,n,ier,r0) ;generate data
20 x=SUBMNS(x,n,1,xbar) ;subtract mean
21 alphayw=DTAR(x,n,p,1,2,1,p,r0,rvaryw) ;use YW
22 alphabg=DTAR(x,n,p,1,2,2,p,r0,rvarbg) ;use Burg
23 LABEL(alphayw)='Yule-Walker Coefficients'
24 LABEL(alphabg)='Burg Coefficients'
25 LIST(alphayw)
26 LIST(alphabg)
27 LIST(rvaryw,rvarbg)
28 fyw=ARSP(alphayw,p,rvaryw,256) ;YW spectra
29 fbg=ARSP(alphabg,p,rvarbg,256) ;Burg spectra
30 LABEL(fyw)='Spectra for Yule-Walker'
31 LABEL(fbg)='Spectra for Burg'
32 PLOTSP(fyw,256,r0,fbg,256,r0) ;plot
33 IF(ns.eq.nsamps,end)
34 ns=ns+1
35 GOTO(s1)
36 ;
37 ;end
38 ;

```

We tested the macro by generating five realizations of length 100 from the AR(4) process described above. The resulting estimates are given in Table 3.11. The Yule-Walker estimates are consistently poor, while the Burg estimates are quite close to the true values. The consistent behavior of the Yule-Walker estimates suggests that the Yule-Walker procedure is trying to estimate the coefficients of some other AR process rather than the AR(4). This is in fact what is happening as the closeness of the zeros of the characteristic polynomial to the unit circle is in essence “tricking” the Yule-Walker procedure into trying to fit an AR(2) model (see Findley (1978) and Quinn (1980) for more discussion of this issue). In Figure 3.18 we superimpose the spectral estimates for the Yule-Walker and Burg estimates for one of the realizations. Note that the

Table 3.11. Example Ouput from the BURGYW Macro

Yule-Walker Coefficients				
1	-1.339443	.843000	.093859	-.031056
Burg Coefficients				
1	-2.644507	3.580996	-2.453292	.864386
rvaryw=15.646350 rvarbg=.889589				
Yule-Walker Coefficients				
1	-1.735767	1.617713	-.571055	.119940
Burg Coefficients				
1	-2.770745	3.834797	-2.690591	.940820
rvaryw=29.172470 rvarbg=.984955				
Yule-Walker Coefficients				
1	-1.132773	.679221	.184523	-.019664
Burg Coefficients				
1	-2.727930	3.748685	-2.601738	.907362
rvaryw=68.174780 rvarbg=1.210733				
Yule-Walker Coefficients				
1	-1.714853	1.457379	-.392310	.064111
Burg Coefficients				
1	-2.759392	3.812833	-2.657903	.928765
rvaryw=22.233480 rvarbg=.897821				
Yule-Walker Coefficients				
1	-1.784269	1.583475	-.514477	.115106
Burg Coefficients				
1	-2.729429	3.654166	-2.452617	.811345
rvaryw=16.139020 rvarbg=.945297				

Yule-Walker estimator fails to “split the peak,” while the Burg estimator seems quite good.

Example 3.13 PREDICTION VARIANCE HORIZONS

The macro PVH.MAC can be used to find the values of PVH for a data set.

```

1 ;;
2 ;;   PVH.MAC: macro to estimate the PVH function
3 ;;
4 ;;   INPUT: x, n (data and sample size)
5 ;;           p, m (AR order to use and maximum index for PVH)
6 ;;
7 ;;   OUTPUT: pvh (array of length m containing PVH(1),...,PVH(m))
8 ;;           pvh0 (real scalar containing PVH(0))
9 PAUSE
10 ;start
11 mpi=m+1
12 rho=CORR(x,n,m,0,1,r0,per)      ;find correlations
13 alpha=CORRAR(rho,r0,p,rvar)     ; and AR parameters
14 gama=INVPOLY(alpha,p,m)        ;find MA(infinity) coefficients

```

```

15 gama=<1,gama>                ;now find PVH
16 gama=gama*gama
17 pvh=CUM(gama,mp1,1)
18 pvh=rvar+pvh
19 pvh=pvh/r0
20 pvh0=pvh[1]
21 pvh=EXTRACT(pvh,2,mp1)        ;finally have it
22 LABEL(pvh)='PVH Function PVH(0)=@pvh0@'
23 LIST(pvh,m,10,10f6.3)

```

Example 3.14 CAT AS A WHITE NOISE TEST

The macro CATWN.MAC will generate a series of white noise realizations of length n and then find the CAT criterion for orders 0 through M , keeping track of how many times each order is chosen.

```

1 ;;
2 ;;   CATWN.MAC: macro to investigate the level of significance
3 ;;               of the CAT criterion as a test for white noise.
4 ;;
5 ;;   INPUT: seed, nsamps (number of samples to generate)
6 ;;           n (sample size)  M (maximum order)
7 ;;
8 PAUSE
9 ;start
10 PROMPTOFF
11 smp#=1           ;initialize sample counter
12 x=WN(seed,10)   ;warm up random number generator
13 ords=LINE(M,0,0) ;reserve space for chosen orders
14 ;
15 ;s1
16 ;
17 LIST(smp#)       ;tell user which sample we're doing
18 x=WN(0,n)
19 x=SUBMWS(x,n,1,xbar)
20 x=DTAR(x,n,M,2,2,1,p,r0,rvar) ;find order
21 IF(p.gt.0)
22     ords[p]=ords[p]+1           ;add 1 to order counter (unless p=0)
23 ENDIF
24 IF(smp#.eq.nsamps,e1)           ;finished?
25     smp#=smp#+1                 ;no
26 GOTO(s1)
27 ;
28 ;e1
29 ;
30 cords=CUM(ords,M,1)
31 nwrong:=cords[M]
32 nright=nsamps-nwrong
33 ;list
34 LIST(n,M)

```

```

35 LABEL(ords)='Number of Times Various Orders Chosen'
36 LIST(ords,M)
37 LIST(nright)
38 PROMPTON

```

To illustrate the macro, we generated 1000 realizations of length 100 and found that 928 of the realizations chose white noise (order 0) when tested only versus order 1. When orders as high as 10 were allowed to be chosen, we found the following frequencies:

Number of Times CAT Chose Order p:

p	0	1	2	3	4	5	6	7	8	9	10
#	827	53	32	31	14	17	9	7	4	4	2

Example 3.15 AIC FOR ARMA PROCESSES

In the ARMAAIC.MAC macro given below, we use ARMASEL to find an estimate of the residual variance for AR orders 1 through pmax and MA orders 1 through qmax. These variances are used to find the corresponding AIC values.

```

1 ;;
2 ;;   ARMAAIC.MAC: macro to find the AIC for ARMA(p,q)
3 ;;               orders p=1,...,pmax, and q=1,...,qmax.
4 ;;
5 ;;   INPUT: x, n (data and sample size)
6 ;;           pmax, qmax
7 ;;
8 ;;   OUTPUT: aicpq (matrix whose (p,q) element is AIC(p,q))
9 ;;           aic00 ( AIC(0,0) )
10 ;;
11 PAUSE
12 ;start
13 pq=pmax*qmax
14 aicpq=LINE(pq,0,0)
15 rho=CORR(x,n,0,0,1,r0,per)
16 aic00=LOGE(r0)
17 aic00=n*aic00
18 p=1
19 q=1
20 PROMPTOFF
21 ii=0
22 ;
23 ;s4
24 ;
25 ii=ii+1
26 alpha=LINE(p,0,1) ;use ARMASEL to estimate residual variance
27 beta=LINE(q,0,1)
28 kopt=p+q

```



```

29 npar2=2*kopt
30 kopt=-kopt
31 mn=-n
32 ARMASEL(x,mn,30,10,p,q,kopt,.95,p,q,alpha,beta,rvar,ier)
33 LIST(alpha,p)
34 LIST(beta,q)
35 aic=LOGE(rvar)
36 aic=n*aic+npar2
37 aicpq[ii]=aic
38 LIST(aic)
39 IF(q.eq.qmax,s5)
40 q=q+1
41 GOTO(s4)
42 ;
43 ;s5
44 ;
45 IF(p.eq.pmax,s6)
46 q=1
47 p=p+1
48 GOTO(s4)
49 ;
50 ;s6
51 ;
52 LABEL(aicpq)='AIC for ARMA(p,q)'
53 LISTM(aicpq,pmax,qmax)
54 LIST(aic00)
55 PROMPTON

```

Example 3.16 MODEL IDENTIFICATION

The basic aim of TIMESLAB is to provide users with enough building blocks so that they can produce very general, easy to use macros for analyzing data. The macro ID.MAC is an example of such a macro. It guides the user through a series of steps that should be helpful in finding transformations and models that can be used with other macros for spectral estimation and prediction.

```

1 CLS ;clear the screen
2 ;;
3 ;; ID.MAC: This macro will guide you through the ARMA model selection
4 ;; process. To use this macro you need to have defined
5 ;; an array x containing your data and the integer n
6 ;; which is the sample size.
7 ;;
8 ;;
9 PAUSE
10 CLS
11 ;; We first try to get a feel for your data. I will plot them as well
12 ;; as their histogram and quantile plot. If either of these appear
13 ;; to be long tailed (i.e. skewed), then a transformation is probably

```

```
14 ;; called for and you should break the macro and consider some power
15 ;; transform.
16 ;;
17 ;; First the plot of the data. It will not have 'nice' tic mark
18 ;; labels.
19 ;;
20 PAUSE
21 PLOT(x,n,0,n,0,0)
22 ;;
23 ;; Now the histogram:
24 ;;
25 PAUSE
26 HIST(x,n)
27 ;;
28 ;; Now the quantile plot. The plot should look linear in the middle
29 ;; and hit the vertical axes near where they intersect the horizontal
30 ;; axes.
31 ;;
32 PAUSE
33 INFQNT(x,n)
34 ;;
35 ;; If those plots looked 'strange,' you should break the macro now
36 ;; and consider transforming the data.
37 ;;
38 PAUSE
39 m=0
40 CLS
41 ;;
42 ;; Now we see if the data could be white noise. We'll plot
43 ;; the sample correlogram with 95% simultaneous confidence
44 ;; bands as well as the cumulative periodogram with its
45 ;; confidence bands.
46 ;;
47 ;; I will use  $m=\min(n/4,30)$  as the number of correlations
48 ;; unless you break the macro, define m, and then resume execution
49 ;; of the macro by entering the command MACRO. I will also use
50 ;; n frequencies for the periodogram.
51 ;;
52 PAUSE
53 CLS
54 ;; If the correlogram falls within the bands, your data may be
55 ;; white noise.
56 ;;
57 ;; If the cumulative periodogram falls completely within
58 ;; the bands, then x might be white noise.
59 ;;
60 ;; If there is an apparent discontinuity in the cumulative
61 ;; periodogram at a certain frequency, then either x might have
62 ;; a deterministic sinusoid of that frequency or x might
63 ;; be an AR process with some roots near the unit circle.
64 ;;
65 PAUSE
66 CLS
67 ;;
```

```

68 ;; This takes a few seconds .....
69 ;;
70 PROMPTOFF
71 IF(m.eq.0) ;determine m
72     m=n/4
73     IF(m.gt.30)
74         m=30
75     ENDIF
76 ENDIF
77 MACRO(wntest)
78 PROMPTON
79 CLS
80 ;;
81 ;; Now we'll see if the semi-automatic methods of ARMA model
82 ;; determination are adequate for your data or if you need to
83 ;; identify a suitable transformation to "stationarity."
84 ;;
85 ;; We will do this by automatically fitting an autoregressive
86 ;; model to the data, plotting its spectral density, and
87 ;; determining the range (max-min) of the logarithm of the
88 ;; spectral density. This range can be used to classify the
89 ;; "memory type" of the series as short (low range) or long
90 ;; (high range). One rule of thumb is to consider transforming
91 ;; the data if the range is greater than 6.
92 ;;
93 ;; Henceforth I will use the next power of 2 greater than or
94 ;; equal to n as the number of frequencies.
95 ;;
96 ;; A large dynamic range can be caused by either an infinity
97 ;; (a peak) or a zero in the spectrum. An infinity can be
98 ;; handled by transformation, while a zero is usually the
99 ;; result of a transformation that caused the data to become
100 ;; a noninvertible MA or ARMA model. If this happens, you
101 ;; might want to undo the offending transformation and perhaps try
102 ;; another.
103 ;;
104 PAUSE
105 CLS
106 ;;
107 ;; This takes a few seconds ....
108 ;;
109 PROMPTOFF
110 ln=LOGE(n) ;find next power of 2 greater than or equal to n
111 le2=LOGE(2)
112 ln={ln/le2}+.99
113 Q:=ln
114 Q=2~Q
115 q={Q/2}+1
116 rho=CORR(x,n,m,0,1,r0,per)
117 alpha=CORRAR(rho,r0,m,n,p,rvar,cat)
118 f=LINE(q,1,0)
119 IF(p.gt.0)
120     f=ARSP(alpha,p,rvar,Q)
121     f=f/r0

```

```

122 ENDDIF
123 ff=LOGE(f,q)
124 MAXMIN(ff,q,fmax,imax,fmin,imin)
125 rge=fmax-fmin
126 LABEL(f)='AR Spectra, p=#p#, range=@rge#'
127 PROMPTON
128 PLOTSP(f,Q,1)
129 ;;
130 ;;   Now we will assume that your data either are short memory or
131 ;;   have been transformed so that they are short memory.
132 ;;
133 ;;   We now look at a variety of diagnostics that will suggest
134 ;;   one or more ARMA type models for your data.
135 ;;
136 ;;   We begin with the sample correlogram. For an MA model this
137 ;;   will be zero past some lag. For an AR or ARMA model it
138 ;;   decays to zero and stays there, or else looks like a damped
139 ;;   sinusoid.
140 ;;
141 PAUSE
142 CLS
143 rho=CORR(x,n,m,0,1,r0,per)
144 zero=LINE(m,0,0)
145 LABEL(zero)=' '
146 PLOT2(rho,zero,m,m,3,3,0,m,-1,1)
147 ;;
148 ;;   Next is the partial correlogram. For an AR process this
149 ;;   will be essentially zero past some lag.
150 ;;
151 PAUSE
152 CLS
153 y=SUBMNS(x,n,1,xbar)
154 ;;
155 ;;   This takes a few seconds .....
156 ;;
157 part=PARCORR(y,n,m,brv)
158 PLOT2(part,zero,m,m,3,3,0,m,-1,1)
159 ;;
160 ;;   Next are the biased residual variances. For an AR process,
161 ;;   these will decrease until the correct order at which point
162 ;;   they remain roughly constant. We will plot the standardized
163 ;;   version of the biased residual variances.
164 ;;
165 PAUSE
166 brv=brv/r0
167 LABEL(brv)='Standardized Biased Residual Variances'
168 PLOT2(brv,zero,m,m,3,3,0,m,0,1)
169 ;;
170 ;;   Next are the standardized unbiased residual variances.
171 ;;   For an AR process, the minimum of this function could
172 ;;   be the AR order.
173 PAUSE
174 nmj=LINE(m,0,1)
175 urv=n*brv/nmj

```

```

176 LABEL(urv)='Standardized Unbiased Residual Variances'
177 PLOT2(urv,zero,m,m,3,3,0,m,0,1)
178 ;inverse
179 ;;
180 ;;   Now we will consider the above quantities for the
181 ;;   process corresponding to the inverse spectral density.
182 ;;
183 ;;   The interpretation of these quantities is the reverse of
184 ;;   those for f.
185 ;;
186 ;;   First the inverse correlogram. Now a truncated correlogram
187 ;;   indicates that x may be an AR process.
188 ;;
189 PAUSE
190 rho=CORR(x,n,m,0,1,r0,per)
191 alpha=CORRAR(rho,r0,m,rvar)
192 rvari=1/rvar
193 rhoi=MACORR(alpha,m,rvari,m,ri0)
194 LABEL(rhoi)='Inverse Autocorrelations'
195 PLOT2(rhoi,zero,m,m,3,3,0,m,-1,1)
196 ;;
197 ;;   Next are the inverse partials, biased residual variances,
198 ;;   and unbiased residual variances:
199 ;;
200 PAUSE
201 alpha=CORRAR(rhoi,ri0,m,rvar)
202 part=ARPART(alpha,m,ier)
203 LABEL(part)='Inverse Partial'
204 PLOT2(part,zero,m,m,3,3,0,m,-1,1)
205 a1=<-1>
206 lpart=1-part^2
207 lpart=log(lpart,m)
208 lpart=<0,lpart>
209 mp1=m+1
210 brv=ARDT(a1,1,mp1,lpart)
211 brv=EXTRACT(brv,2,mp1)
212 brv=EXP(brv,m)
213 LABEL(brv)='Inverse Biased Residual Variances'
214 PLOT2(brv,zero,m,m,3,3,0,m,0,1)
215 urv=n*brv/nmj
216 LABEL(urv)='Inverse Unbiased Residual Variances'
217 PLOT2(urv,zero,m,m,3,3,0,m,0,1)
218 ;;
219 ;;   Now we use the AIC to look for the best full
220 ;;   AR model:
221 ;;
222 ;;
223 ;aic
224 PAUSE
225 y=SUBMNS(x,n,1,xbar)
226 ;;
227 ;;   This takes a few seconds .....
228 ;;
229 part=PARCORR(y,n,m,brv)

```

```

230 aicar=loge(brv,m)
231 aicar=aicar*n
232 ajj=Line(m,0,2)
233 aicar=aicar+ajj
234 LABEL(aicar)='AIC'
235 aic0=LOGE(r0)
236 aic0=n*aic0
237 CLS
238 LIST(aicar)
239 LIST(aic0)
240 PAUSE
241 ;armasel
242 CLS
243 ;;
244 ;;   Now ARMASEL to look for Subset AR. We'll do it twice,
245 ;;   the first time allowing the possibility of more lags being
246 ;;   included than the second time.
247 ;;
248 ;;   pval is .5 for the first one
249 ;;
250 PAUSE
251 PROMPTOFF
252 s=10
253 k1=m-s
254 k2=0
255 rvar=r0
256 ARMASEL(y,n,m,s,k1,k2,0,.5,p,q,alpha,beta,rvar,ier)
257 npar2=0
258 IF(p.gt.0)
259     LIST(alpha,p)
260     aa=EXTRACT(alpha,p,0,eq,npar2)
261     npar2=p-npar2
262     npar2=2*npar2
263 ENDIF
264 aic1=loge(rvar)
265 aic1=n*aic1+npar2
266 LIST(aic1)
267 PROMPTON
268 ;;
269 ;;   Now the second time (pval=.95):
270 ;;
271 PAUSE
272 PROMPTOFF
273 rvar=r0
274 ARMASEL(y,n,m,s,k1,k2,0,.95,p,q,alpha,beta,rvar,ier)
275 npar2=0
276 IF(p.gt.0)
277     LIST(alpha,p)
278     aa=EXTRACT(alpha,p,0,eq,npar2)
279     npar2=p-npar2
280     npar2=2*npar2
281 ENDIF
282 aic1=loge(rvar)
283 aic1=n*aic1+npar2

```



```
284 LIST(aic1)
285 PROMPTON
286 PAUSE
287 ;armaaic
288 CLS
289 ;;
290 ;;   Now we will look for ARMA models:
291 ;;
292 ;;   First we use ARMASEL to get AIC for each combination of p and q
293 ;;   between 1 and 5
294 ;;
295 ;;   This takes a while and you will see each model.
296 ;;
297 PAUSE
298 PROMPTOFF
299 pmax=5
300 qmax=5
301 MACRO(armaaic,start)
302 PROMPTON
303 PAUSE
304 ;gpac
305 CLS
306 ;;
307 ;;   Now we find the GPAC array for maximum AR and MA orders
308 ;;   being 7. This also takes a while.
309 ;;
310 PAUSE
311 PROMPTOFF
312 pmax=7
313 qmax=7
314 MACRO(GPAC)
315 PROMPTON
316 PAUSE
317 ;armasub
318 CLS
319 ;;
320 ;;   Finally, we will let ARMASEL suggest a subset ARMA model
321 ;;   using pval=.5
322 ;;
323 PAUSE
324 PROMPTOFF
325 rvar=r0
326 mm10=m-10
327 ARMASEL(y,n,m,10,mm10,mm10,0,.5,p,q,alpha,beta,rvar,ier)
328 LIST(p,q)
329 npar2=0
330 IF(p,6,6,1)
331 aa=EXTRACT(alpha,p,0,eq,np2)
332 np2=p-np2
333 np2=2*np2
334 npar2=npar2+np2
335 LIST(alpha,p)
336 IF(q,6,6,1)
337 aa=EXTRACT(beta,q,0,eq,np2)
```

```

338 np2=q-np2
339 np2=2*np2
340 npar2=npar2+np2
341 LIST(beta,q)
342 aic1=loge(rvar)
343 aic1=n+aic1
344 aic1=aic1+npar2
345 LIST(aic1)
346 PROMPTON
347 PAUSE
348 CLS
349 ;; This is the end of the model identification macro.

```

Example 3.17 FISHER'S EXACT TEST

Given a data set, the macro FISHER.MAC given below will return the value of the test statistic, the frequency for which it occurs, and the p -value for the test statistic. It calls the macro FISHPV.MAC to find the p -value.

```

1 ;;
2 ;; FISHER.MAC: Macro to perform Fisher's exact test for
3 ;; the presence of a deterministic sinusoid.
4 ;;
5 ;; INPUT: x,n
6 ;;
7 ;; OUPUT: g (value of statistic), freq (peak frequency),
8 ;; pval (pvalue of g).
9 ;;
10 PAUSE
11 ;start
12 y=SUBMMS(x,n,1,xbar)
13 q={n/2}+1
14 FFT(y,n,q,1,zr,zi)
15 per={zr^2+zi^2}/n
16 MAXMIN(per,q,g,imax,fmin,imin) ;g is now max(per)
17 freq={imax-1.}/n ;now we've got freq
18 per=CUM(per,q,1) ;crude way to get denominator
19 g=g/per[q] ;now we've got g
20 MACRO(fishpv,start) ;find p-value
21 LIST(pval,freq)

1 ;;
2 ;; FISHCDF.MAC: Macro to find p-value of Fisher's exact test for
3 ;; specified n and g
4 ;;
5 ;; INPUT: n,g
6 ;;
7 ;; OUPUT: pval
8 ;;

```

```

9 PAUSE
10 ;start
11 m=n/2
12 k:=1/g          ;k is [1/g]
13 LIST(g,k)
14 ;
15 ; Find p-value:
16 ;
17 ;;
18 ;; Calculating p-value ...
19 ;;
20 PROMPTOFF
21 j=1
22 pval=0.0          ;initialize pval
23 ;
24 ;startloop
25 ;
26 LIST(j)
27 pp=BINOM(m,j)
28 pval=pval+{{-1.}}^{j-1}}*pp*(1-j*g)^{m-1}
29 IF(j.eq.m,endoloop)
30 IF(j.eq.k,endoloop)
31 j=j+1
32 GOTO(startloop)
33 ;
34 ;endoloop
35 ;
36 PROMPTON

```

Example 3.18 CONFIDENCE BANDS FOR AR SPECTRA

The macro LYNXCB.MAC was used to form Figure 3.14. It is written specifically for the lynx data but can be easily modified for other data sets.

```

1 ;;
2 ;; LYNXCB.MAC: macro to find confidence bands for the spectral
3 ;; density for the lynx data
4 ;;
5 ;; INPUT: conf (level of confidence)
6 ;;
7 PAUSE
8 ;start
9 READ(lynx,x,n)
10 ;start1          ;entry point if we already read data
11 y=LOGE(x,n)
12 ee=LOGE(10)
13 y=y/ee          ;now y has log10 of lynx data
14 y=SUBMNS(y,n,1,ybar)
15 alpha=DTAR(y,n,20,2,2,1,p,r0,rvar)      ;use CAT to choose order
16 f=ARSP(alpha,p,rvar,128)                ;point estimate
17 ARSPCB(alpha,p,rvar,n,128,conf,fl,fu)    ;find bands

```

```

18 ;
19 ; Note how we use LABEL command. In PLOTSP, the label
20 ; for the last array goes on top.
21 ;
22 nc:=100*conf
23 LABEL(f1)='AR spectral estimator and #nc#% bands'
24 LABEL(f)='for lynx data'
25 LABEL(fu)=' '
26 PLOTSP(f,128,r0,f1,128,r0,fu,128,r0)

```

Example 3.19 ESTIMATING PEAK FREQUENCIES

The macro LYNXPEAK.MAC was used to form Figure 3.16.

```

1 ;;
2 ;; LYNXPEAK: macro to find the confidence interval for the
3 ;; period for the lynx data and produce plots of
4 ;; the log10 of the data and the AR(11) spectra
5 ;; with the values for the confidence interval
6 ;; placed on the plot.
7 ;;
8 ;; INPUT: none
9 ;;
10 PAUSE
11 READ(lynx,x,n)
12 y=LOGE(x,n)
13 ee=LOGE(10)
14 y=y/ee
15 year=LINE(n,1820,1) ;this gets array 1821,...,1934
16 LABEL(y)='log10 of lynx data'
17 LABEL(year)=' '
18 PLOT(year,y,n,1820,1940,1,4)
19 y=SUBMNS(y,n,1,ybar) ;subtract mean
20 alpha=DTAR(y,n,11,1,2,1,p,r0,rvar) ;fit AR(11)
21 ARSPPEAK(alpha,p,rvar,n,ier,freq,se) ;find peak frequency and std error
22 se2=2.*se
23 fup=freq+se2
24 flow=freq-se2 ;flow and fup are limits on frequency
25 plow=1./fup
26 pup=1./flow ;plow and pup are limits on period
27 f=ARSP(alpha,p,rvar,256) ;AR(11) spectral density
28 LABEL(f)='Lynx data, Period CI: (@plow@, @pup@)' ;form plotting label
29 PLOTSP(f,256,r0)

```

Example 3.20 AUTOREGRESSIVE SPECTRAL ESTIMATION

The macro ARSP.MAC will find the autoregressive spectral estimator for a data set using either a specified order or the order that the CAT criterion chooses. This macro was used to form Figure 3.13.

```

1 ;;
2 ;;   ARSP.MAC: macro to perform autoregressive spectral estimation.
3 ;;
4 ;;   INPUT: x, n (data and sample size)
5 ;;           iopt ( 1 means fit order m, 2 means fit CAT order .le. m)
6 ;;           m (order or maximum order to use)
7 ;;           Q (number of frequencies)
8 ;;
9 PAUSE
10 ;start
11 y=SUBMNS(x,n,1,xbar)
12 alpha=DTAR(y,n,m,iopt,1,1,p,r0,rvar,cat)
13 IF(p.eq.0,end)
14 f=ARSP(alpha,p,rvar,Q)
15 LABEL(f)=<x>
16 PLOTSP(f,Q,r0)
17 ;end

```

Example 3.21 PROPERTIES OF AR SPECTRAL ESTIMATION

The macro ARSPSM.MAC generates a series of realizations from a user-specified autoregressive process and superimposes on one set of axes the autoregressive spectral estimates from each realization. Using this macro for a variety of types of processes can give insight into the behavior of the estimator.

```

1 ;;
2 ;;   ARSPSM.MAC: macro to illustrate sampling properties of
3 ;;               AR spectral estimation.
4 ;;
5 ;;               The AR spectral estimates for nsamps samples
6 ;;               of size n are superimposed.
7 ;;
8 ;;   INPUT: p, alpha (true order and coefficients)
9 ;;           nsamps, n, maxp (number of samples, seed, maximum order)
10 ;;           seed (random number generator seed)
11 ;;
12 PAUSE
13 ;start
14 PLOTOM                ;switch to graphics mode
15 PLOTSIZE(0)           ;use whole screen for graphs
16 BATCHOM              ;don't pause between graphs
17 x=WN(seed,10)         ;warm up generator
18 ns=1                  ;initialize counter
19 ;
20 ;startloop
21 ;
22 x=ARDT(alpha,p,1,0,n,ier,r0)           ;generate data
23 x=SUBMNS(x,n,1,xbar)                   ;subtract mean
24 alpha1=DTAR(x,n,maxp,2,2,1,p1,r0,rvar) ;fit AR
25 f=ARSP(alpha1,p1,rvar,256)             ;find spectra

```

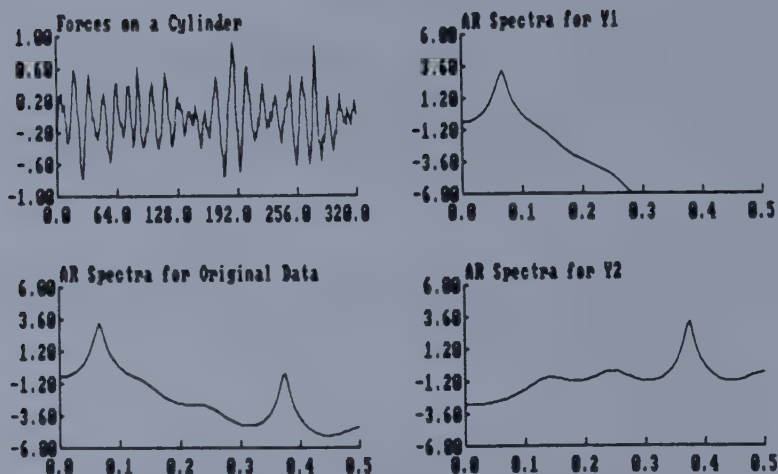


Figure 3.19. The Results of the Autoregressive Filtering Example.

```

26 LABEL(f)='Sample #ns# of #nsamps#'          ;tell user which sample
27 PLOTSP(f,256,r0)
28 IF(ns.eq.nsamps,endoloop)                    ;done all samples?
29 ns=ns+1                                       ;no
30 GOTO(startloop)
31 ;
32 ;endloop                                     ;yes
33 ;
34 GRMENU(0,.5,-6,6)
35 BATCHOFF
36 PLOTOFF

```

Example 3.22 AUTOREGRESSIVE FILTERING

We saw in the bandpass filter example in Section 2.3 that it is possible to remove frequency components from a time series by applying what is essentially a moving average smoother to the data. This smoother can be thought of as a nonparametric method as it uses none of the properties of the data and assumes no model for it. We also saw that such smoothers can lead to distortion of the spectral density in other frequency ranges.

One solution to this problem is to apply an autoregressive filter to the data. For example, consider the wave data in Figure 3.19. This data set consists of the forces on a cylinder suspended in a tank of water. The water in the tank has waves with period approximately 2 seconds. The sampling interval for the forces recorded on the cylinder is 0.15 second. Thus we would expect to see

a periodicity in the data of approximately 15 time periods. Unfortunately, the instrument used to measure the forces on the cylinder induced some high frequency oscillation onto the plot of the data. Thus the data appear to be fairly regular in their 2-second oscillation, but also have a rapid oscillation. The researchers analyzing the data removed this high frequency oscillation using a low pass filter but were dissatisfied with the amount of distortion caused by the filter.

We applied the ARSP macro with `iopt=2` and `m=20` and found that an AR(10) process with the coefficients given in Table 3.12 best fit the data. We also give the roots of the AR(10) polynomial (as found by the POLYROOTS command) and their moduli in Table 3.12.

Table 3.12. AR Coefficients and Roots for the Wave Data

Autoregressive Coefficients				
-1.24154	.429067	-.078211	.455949	-.463727
.337943	.099405	-.397343	.445400	-.159155
Real Part of Polynomial Roots				
-1.23660	-.723344	-.723344	.04786	.04786
.82552	.825529	.947287	.94728	1.84047
Imaginary Part of Polynomial Roots				
.000000	.734836	-.734836	-1.23346	1.23348
.951482	-.951482	-.420146	.420146	.00000
Moduli of Roots				
1.23660	1.03112	1.03112	1.23441	1.23441
1.25969	1.25969	1.03628	1.03628	1.84047

In Figure 3.19 we give the resulting autoregressive spectral estimator. Notice that it clearly demonstrates the wave oscillation and high frequency noise contained in the data. Inspection of the roots shows that two of the complex pairs are close to the unit circle, the first being roots 2 and 3, and the second being roots 8 and 9. Thus we formed two sets of AR(2) coefficients via the ROOTSPOLY command and formed a time series Y_1 by applying the first AR(2) filter to the original data, and a series Y_2 by applying the second AR(2) filter to the original data (the ARFILT command was used with `rvar=1`). These two series were then each analyzed by the ARSP. Order eight was chosen in each case. The resulting autoregressive spectral estimates are given in Figure 3.19 which shows that indeed we have identified the parts of the model corresponding to the two peaks in the original spectral density. Notice how little distortion is evident in the spectral densities of the filtered data sets.

Computational Problems

C3.1. For a realization of length 100 from the AR(2) process with coefficients 0.3 and 0.9 and error variance $\sigma^2 = 1$, what is the equivalent number of independent observations for estimating μ ? (Hint: Use `ARCORR` and `ARSP` to find $R(0)$ and $f(0)$, respectively.)

C3.2. Theorem 3.1.1 shows that if X is a covariance stationary time series with summable autocovariance function, then the variance of \bar{X}_n goes to zero at the rate of $1/n$; that is, for large n , $n\text{Var}(\bar{X}_n)$ is roughly constant, with limiting value equal to $f(0)$ if the spectral density f of X exists and is continuous at frequency 0. Tsay (1986) shows that if X is a random walk process with independent, identically distributed errors ϵ , then the variance of \bar{X}_n in fact goes to infinity at the rate of n ; that is, $n^{-1}\text{Var}(\bar{X}_n)$ is roughly constant, with limiting value $\sigma^2/3$, where σ^2 is the variance of the error series ϵ . Verify this fact by generating 500 realizations from a Gaussian random walk having $\sigma^2 = 1$ for each of sample sizes $n = 100, 200$, and 400 , calculating the sample variance for each of the three sets of 500 sample means, and then dividing each of these sample variances by n . Each of these three estimates of $n^{-1}\text{Var}(\bar{X}_n)$ should be approximately $1/3$. (Hint: See the `MEDMN` macro for an example of writing a simulation macro.)

C3.3. The Bartlett test for white noise relies on the fact that the cumulative periodogram of white noise behaves like the ordered values of a sample from a $U(0,1)$ population. Write a macro that will generate `nsamps` samples of size `n` from a $U(0,1)$ population and superimpose the plot of the ordered values of the samples on the same axes. Use the `SORT` command to find the ordered values. Have the values of `nsamps` and `n` be input to the macro and make sure that the vertical axis ranges from zero to one. Try the macro for `nsamps=100` and `n=100`. Roughly speaking the resulting graph consists of a set of bands within which a uniform white noise series should fall. Is the band of roughly constant width on the graph?

C3.4. What are the p -values for the test of white noise for the rainfall data (Series IV in Section 1.1) by the `BARTTEST` and `QTEST` commands? Do they reach the same conclusion about whether the rainfall data come from a white noise process?

C3.5. Smooth the periodogram of the artificial data set described in Section 1.1 using the `AVEPER` macro with 1, 2, and 3 for `m`. What happens to the peak in the periodogram at frequency 0.10 as `m` increases?

C3.6. Evaluate the lag window generators for the Bohman and Tukey windows at the 101 equally spaced values between 0 and 1 inclusive, and use the **PLOTK** command to superimpose the plots of the two generators versus the values at which they were evaluated. Use a line plot for the Tukey window generator and a point plot (with lower triangles as the plotting symbol) for the Bohman window generator. Which generator seems to decay faster in the neighborhood of $u = 0$?

C3.7. Use the **ARMASEL** command to choose a subset AR model for the Wolfer sunspot data and then use the output of **ARMASEL** as the input to the **SEASEST** command to obtain maximum likelihood estimates of the coefficients of the model. Finally, use the **SEASPREL** command to find predictors and prediction limits for the next 24 values of the series.

C3.8. Generate five realizations of length 100 from the AR(4) process discussed in Example 3.12 and use the **LSAR** macro to find parameter estimates for each realization. Are the results more similar to the Burg estimates or to the Yule-Walker estimates found in Example 3.12?

C3.9. Apply the **ARSPPEAK** command to the Wolfer sunspot data in an attempt to find a confidence interval for the so-called sunspot cycle. Use the subset AR model chosen in Problem C3.7.

C3.10. Use the **MSERHO** macro with any AR(3) model of your choice. Do the results behave similarly to those in Example 3.2?

C3.11. Perform the experiment described in Section 3.2.1 and see if you get similar results.

C3.12. Generate a sample of size 200 from an MA(1) process having $\beta = 2$ and $\sigma^2 = 1$ and then use the **INNOV** macro to estimate σ^2 . Explain why this estimate is close to 4 instead of 1. (Hint: See the discussion of nonidentifiability of MA processes in Section 2.5.3.)

C3.13. Find and plot $f''(\omega)$ for the AR(2) process having coefficients .3 and .9 and error variance $\sigma^2 = 1$. (Hint: The second derivative of the reciprocal of f is easy to evaluate; see Section 3.9.2.)

C3.14. Show that the AIC chooses order 11 for the log (base 10) of the lynx data (use the **AIC** macro), while the error variance estimate obtained from **ARMASEL** for the subset model having lags 1, 2, 4, 10, and 11 has a smaller AIC value than any of those of the full AR models.

C3.15. How large must n be before $\log \log n$ is greater than or equal to 3? The log here refers to the natural log.

C3.16. Use the `PLOT2` command to superimpose the graphs of $\log n/n$ and $n^{1/5}/n$ for $n = 10, \dots, 100$. Have the horizontal axis range from 10 to 100 and the vertical axis range from 0 to 0.25. What do these two functions look like for $n = 1, \dots, 10$?

C3.17. Use the `ARSPPEAK` command to find the location of the peak frequency in the true $\text{AR}(5)$ spectral density in Figure 3.5.

C3.18. Use the `PVH.MAC` macro to find and graph the PVH for the first and 12th difference of the log of the sales data. What does this tell us about whether an $\text{MA}(11)$ or $\text{MA}(12)$ lag should be used in the model for these data?

C3.19. Write a macro that will generate a user-specified number of realizations of a user-specified length from a Gaussian white noise process. For each realization, use the AIC to determine if order 0 or 1 is appropriate (be clever here as how long the macro takes to finish depends on how you do this part). Test the macro by generating 500 realizations of length 100. What proportion of the time is order 1 (incorrectly) chosen? How does this compare with the asymptotic result given in Problem T3.14?

C3.20. What does the spectral density of the $\text{AR}(12)$ process having $\alpha_{12} = -.9$ and all other coefficients zero look like?

C3.21. Find the first 20 partial autocorrelations of an $\text{MA}(1)$ process having coefficient .7. Is there any pattern in the partials?

C3.22. Use the `ID` macro for the data set that you are analyzing from your field of interest. Were any preliminary transformations necessary? What model did you find for the data?

C3.23. Use the `ARSPPEAK` command to estimate the peak frequency in the spectral density of each of the two LH time series. Use the standard errors of the two estimators to test whether the two frequencies are significantly different.

C3.24. Use the `ARSP` macro described in Example 3.20 to show that there is no clear peak at the sunspot cycle frequency in the critical radio frequencies data

for $p = 14$.

Theoretical Problems

T3.1. Let \bar{X}_n be the sample mean of a realization $\mathbf{X}_n = (X(1), \dots, X(n))^T$ of length n from a covariance stationary time series X having autocovariance function R .

a) Show that $\bar{X}_n = \mathbf{h}_n^T \mathbf{X}_n$, where $\mathbf{h}_n = (1/n, \dots, 1/n)^T$.

b) Use Theorem A.4.1 to show that

$$\text{Var}(\bar{X}_n) = \frac{1}{n^2} \sum_{j=1}^n \sum_{k=1}^n R(|j-k|)$$

and thus show that

$$\begin{aligned} n \text{Var}(\bar{X}_n) &= \sum_{v=-n}^n \left(1 - \frac{|v|}{n}\right) R(v) \\ &= 2T_n - R(0), \end{aligned}$$

where

$$T_n = \sum_{v=0}^n \left(1 - \frac{|v|}{n}\right) R(v).$$

c) For $n \geq 0$, the n th partial sum of the sequence $R(0), R(1), \dots$ is given by

$$S_n = \sum_{v=0}^{n-1} R(v),$$

while for $n \geq 1$, the n th Cesàro sum is given by the average of the first n partial sums, that is, by

$$C_n = \frac{1}{n} \sum_{j=0}^{n-1} S_j.$$

Show that $C_n = T_n$.

d) It can be shown (see Anderson (1971), p. 460 for example) that a summable sequence is also Cesàro summable and that the two sums are the same; that is, if $\lim_{n \rightarrow \infty} S_n < \infty$, then $\lim_{n \rightarrow \infty} C_n < \infty$ and the two limits are

equal. Use this fact and parts (a), (b), and (c) above to finish the proof that if R is summable, then

$$\lim_{n \rightarrow \infty} n \text{Var}(\bar{X}_n) = \sum_{v=-\infty}^{\infty} R(v).$$

T3.2. If the coefficient of an AR(1) process having mean μ is very close to one, why do we need only two observations in order to have a very good idea of what μ is? This phenomenon is an example of what are called “antithetic variables” (see Fishman (1973)).

T3.3. Let $\{X_n, n = 1, 2, \dots\}$ be a sequence of random variables. Use the continuous function theorem to show that if $X_n \sim \text{AN}(\mu, \mu^2)$, then $\log X_n \sim \text{AN}(\log \mu, c)$, where c does not depend on μ .

T3.4. Show that if X has a χ^2_2 distribution, then $Y = \frac{1}{2}X$ has the exponential distribution with mean one.

T3.5. Let $X(1), \dots, X(n)$ be a realization from a Gaussian $\text{WN}(\sigma^2)$ process. Let ω_j and ω_k be two different natural frequencies, neither of which is 0 or .5. Use Theorem 1.4.1 and parts (b), (e), and (f) of Theorem A.4.2 to show that $2\hat{f}(\omega_j)/\sigma^2$ and $2\hat{f}(\omega_k)/\sigma^2$ are independent, identically distributed as χ^2_2 variables.

T3.6. Show that the unbiased estimate \tilde{R} is not positive definite for the realization $\mathbf{X} = (1/2, -1/2)^T$. (Hint: Show that the matrix $\text{TOEPL}(\tilde{R}(0), \tilde{R}(1))$ is not positive definite.)

T3.7. Use part (c) of Theorem 3.1.3 to show that for an AR(1) process,

$$\lim_{n \rightarrow \infty} n \text{Cov}(\hat{\rho}(1), \hat{\rho}(2)) = 2\rho(1 - \rho^2),$$

where $\rho = -\alpha$.

T3.8. Let X be an ARMA process. Show that the coefficients γ of the $\text{MA}(\infty)$ representation of X satisfy a difference equation. Thus conclude that they are bounded by a geometrically decreasing sequence, and thus so do the corresponding autocovariances. Finally, use these facts to show that the spectral density is p -differentiable for all integers p .

T3.9. Show for a Gaussian AR(1) process having coefficient α that

$$\text{Var}(\hat{R}(0)) \doteq \frac{2R^2(0)}{n} \frac{1 + \alpha^2}{1 - \alpha^2}.$$

From this, find an expression for the equivalent number of uncorrelated observations for estimating $R(0)$. (Hint: The asymptotic variance of the sample variance for a random sample of size N from a population having variance $R(0)$ is $2R^2(0)/N$.)

T3.10. Show that if the lag window $k_n(v)$ is an even function of v , then the corresponding spectral window is real and periodic of period 1 and is symmetric about 0.

T3.11. Let $\tilde{f}_T(\omega)$ and $\tilde{f}_{TP}(\omega)$ denote nonparametric spectral estimators using the same truncation points and the Tukey and truncated periodogram windows respectively.

a) Show that

$$\tilde{f}_T(\omega) = 0.54\tilde{f}_{TP}(\omega) + 0.23\tilde{f}_{TP}\left(\omega - \frac{1}{2M}\right) + 0.23\tilde{f}_{TP}\left(\omega + \frac{1}{2M}\right).$$

b) Thus show that if we calculate the two estimators at the frequencies $\omega_j = (j-1)/Q$, for $j = 1, \dots, [Q/2] + 1$, and $2M$ divides Q , then the Tukey estimator at frequency ω_j is a weighted average of the truncated periodogram estimator at ω_j and the frequencies $Q/2M$ before and after ω_j . Thus if $2M = Q$, the Tukey estimator at a certain frequency is the weighted average of the truncated periodogram at that frequency and the frequencies on either side.

T3.12. Let $X(1), \dots, X(n)$ be a realization of length n from a zero mean, covariance stationary time series X which has a spectral density f . Suppose that $\bar{X} = 0$, and consider dividing the data into K possibly overlapping segments, each of length L , with the starting points of consecutive segments being D time units apart:

$$X_1(t) = X(t)$$

$$X_2(t) = X(D + t)$$

$$\vdots$$

$$X_K(t) = X((K-1)D + t),$$

for $t = 1, \dots, L$ and $(K-1)D + L = n$. Define the K series Y_1, \dots, Y_K , each of length L , by

$$Y_j(t) = W(t)X_j(t), \quad t = 1, \dots, L,$$

where $W(1), \dots, W(L)$ is a weight function called a data window. Next, define the modified periodogram \tilde{f}_j of $Y_j(1), \dots, Y_j(L)$ by

$$\tilde{f}_j(\omega_k) = \frac{1}{UL} \left| \sum_{t=1}^L Y_j(t) e^{-2\pi i(t-1)\omega_k} \right|^2,$$

where $U = \frac{1}{L} \sum_{t=1}^L W^2(t)$, at the frequencies $\omega_k = (k-1)/L$, for $k = 1, \dots, [L/2] + 1$, and form the spectral estimator \tilde{f} of f by the average of these modified periodograms of the K segments:

$$\tilde{f}(\omega_k) = \frac{1}{K} \sum_{j=1}^K \tilde{f}_j(\omega_k), \quad k = 1, \dots, [L/2] + 1.$$

Note that this method of spectral estimation requires only the Fourier transform of segments of the data and not the entire data set (see Welch (1967) for more information about this method).

a) If $L = D$ and $n = KL$, that is, the segments are not overlapping, and $W(t) = 1$ for all t , then show that

$$\tilde{f}(\omega_k) = \sum_{v=-L}^L \left(1 - \frac{|v|}{L}\right) \tilde{R}(v) e^{-2\pi i v \omega_k};$$

that is, the resulting estimator is the same as that obtained using the Bartlett window except that the unbiased sample autocovariances are used. Note that this is the original motivation that Bartlett (1950) used in proposing his estimator, that is, to estimate f by the average of periodograms over nonoverlapping subsets of the data.

b) Show that if

$$W(t) = 1 - \left| \frac{(t-1) - \frac{L-1}{2}}{\frac{L+1}{2}} \right|, \quad t = 1, \dots, L,$$

then $\tilde{f}(\omega_k)$ corresponds to a window estimator whose spectral window is approximately the same as the Parzen window.

T3.13. Let \mathbf{x} and \mathbf{y} be two n -dimensional vectors. Show that the value of a that minimizes

$$S(a) = \sum_{i=1}^n (x_i - ay_i)^2 + \sum_{i=1}^n (y_i - ax_i)^2$$

is given by

$$\hat{a} = \frac{\sum_{i=1}^n x_i y_i}{2 \left[\sum_{i=1}^n x_i^2 + \sum_{i=1}^n y_i^2 \right]}.$$

T3.14. Let $X(1), \dots, X(n)$ be a realization of length n from a Gaussian white noise process.

a) Show that $\text{AIC}(0)$ is less than $\text{AIC}(1)$ if and only if

$$n\hat{\rho}^2 < n \left(1 - e^{-2/n} \right),$$

where $\hat{\rho}^2$ is the sample autocorrelation coefficient of lag one.

b) Thus for large n , show that using AIC as a test for white noise versus the alternative of first-order serial correlation has Type I error probability 0.8427. (Hint: Show that $n\hat{\rho}^2 \sim \chi_1^2$ and that $\lim_{n \rightarrow \infty} n(1 - e^{-2/n}) = 2$. You can find $\Pr(\chi_1^2 < 2)$ via the **DIST** command.)

T3.15. Let $X \sim \text{AR}(\infty)$ whose coefficients satisfy $\alpha_j \leq c\rho^k$ for some $\rho \in [-1, 1]$. Let $p_n = \log n$. Show that conditions (i), (ii), and (iii) in part (a) of Theorem 3.8.3 are satisfied for this choice of p .

T3.16. If we let $g_{(r)}$ denote the the r th largest value of the cumulative periodogram, then Grenander and Rosenblatt (1957) have shown under the null hypothesis that X is Gaussian white noise that

$$\Pr(g_{(r)} > z) = \frac{n!}{(r-1)!} \sum_{j=1}^a \frac{(-1)^{j-r} (1-jz)^{n-1}}{j(n-j)!(j-r)!},$$

where $a = [1/z]$. Show that this reduces to Fisher's exact test when $r = 1$.

T3.17. Find the coefficients of the (nonstationary) ARMA model for X in the model

$$g(L)G(L)W(t) = h(L)H(L)\epsilon(t),$$

where $W(t) = (1 - L)^2 X(t)$, $g(L) = 1$, $G(L) = 1 + .5L^2$, $h(L) = 1 - .7L$, and $H(L) = 1$.

T3.18. Suppose that X is a zero mean, covariance stationary time series with autocovariance function R_X . Suppose that we do not observe X directly, but rather an “amplitude modulated” (see Parzen (1963b)) version of it:

$$Y(t) = g(t)X(t),$$

where g is a bounded, nonrandom function such that

$$R_g(v) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{t=1}^{n-v} g(t)g(t+v)$$

exists for all $v \in \mathcal{Z}$.

a) Show that Y is not covariance stationary but is asymptotically covariance stationary in the sense that if we define

$$\hat{R}_Y(v) = \frac{1}{n} \sum_{t=1}^{n-v} Y(t)Y(t+v),$$

then $\lim_{n \rightarrow \infty} E(\hat{R}_Y(v))$ exists and is given by

$$R_Y(v) = R_g(v)R_X(v).$$

Note that Parzen (1963b) has shown that for any lag v for which $R_g(v) \neq 0$, $\hat{R}_Y(v)/R_g(v)$ is a consistent estimator of $R_X(v)$.

b) Now suppose that we have a realization $X(1), \dots, X(n)$ from X , and that some of the data points are missing. Define

$$g(t) = \begin{cases} 1, & \text{if } X(t) \text{ is not missing} \\ 0, & \text{if } X(t) \text{ is missing.} \end{cases}$$

Consider the case of systematically missing values; that is, we observe m values, then k are missing, then we observe m more, and so on. Note that this situation arises in any time series that periodically cannot be observed, for example, a medical data set where observations are not made on a patient overnight. Show

that if $m > k$, that is, more data are observed than are missed, then R_g is periodic of period $m + k$, and

$$R_g(v) = \frac{\max(v, m) - \min(k, v)}{m + k}, \quad v = 0, 1, \dots, m + k.$$

c) Now suppose that we have missing data and $g(1), \dots, g(n)$ are iid binary random variables, that is, $\Pr(g(i) = 1) = p$ and $\Pr(g(i) = 0) = q = 1 - p$, and the g 's are independent of the X 's. Show that if X is a Gaussian process, then

$$\hat{R}_X(v) = \frac{\hat{R}_Y(v)}{\hat{R}_g(v)}$$

is a consistent estimator of $R_X(v)$, where

$$\hat{R}_g(v) = \frac{1}{n} \sum_{t=1}^{n-v} g(t)g(t+v), \quad v = 0, 1, \dots, n-1.$$

(Hint: The weak law of large numbers can be used to show that $\hat{R}_g(v) \xrightarrow{\mathcal{P}} p^2$, while Isserlis's formula (see above Theorem 3.1.3) can be used to show that $\hat{R}_Y(v) \xrightarrow{m.s.} p^2 R_X(v)$, which means that $\hat{R}_Y(v) \xrightarrow{\mathcal{P}} p^2 R_X(v)$ since convergence in mean square implies convergence in probability.) Given consistent estimators of R_X , we can use them to do nonparametric spectral estimation, to fit ARMA models, and so on.

CHAPTER 4

Analyzing Bivariate Time Series

In this chapter we consider the analysis of data from a bivariate time series $\{\mathbf{X}(t), t \in \mathcal{Z}\}$. Thus at each time point, we have observations $X_1(t)$ and $X_2(t)$ on two phenomena, and we write $\mathbf{X}(t) = (X_1(t), X_2(t))^T$. Sometimes we will think of the data as a single realization from a vector time series, and at other times we will consider that we have a pair of univariate data sets. We will try to emphasize mostly the ideas and methods that are new to the bivariate situation, particularly in the problem of determining what information is contained in one of the univariate series that helps to explain the behavior of the other.

4.1. Probability Theory for Bivariate Series

We begin our discussion by considering what is meant by bivariate covariance stationarity.

4.1.1. Covariance Stationarity

Definition. The bivariate time series $\{\mathbf{X}(t), t \in \mathcal{Z}\}$ is said to be covariance stationary if the univariate series $\{X_1(t), t \in \mathcal{Z}\}$ and $\{X_2(t), t \in \mathcal{Z}\}$ are covariance stationary in the univariate sense, and there exists a sequence $\{R_{12}(v), v \in \mathcal{Z}\}$ such that

$$\text{Cov}(X_1(t), X_2(t+v)) = R_{12}(v), \quad \text{for all } t.$$

The sequence R_{12} is called the cross-covariance sequence of \mathbf{X} . We denote the autocovariances of X_1 and X_2 by R_{11} and R_{22} , respectively.

Note that the cross-covariance sequence is not symmetric about zero, that is, $R_{12}(v) \neq R_{12}(-v)$, but rather

$$R_{12}(v) = \text{Cov}(X_1(t), X_2(t+v)) = \text{Cov}(X_2(t+v), X_1(t)) = R_{21}(-v).$$

If we define the (2×2) matrix

$$\mathbf{R}(v) = \begin{bmatrix} R_{11}(v) & R_{12}(v) \\ R_{21}(v) & R_{22}(v) \end{bmatrix}, \quad v \in \mathcal{Z},$$

we have that $\mathbf{R}(v) = \text{Cov}(\mathbf{X}(t), \mathbf{X}(t+v))$, which can thus be regarded as the autocovariance function of the bivariate series \mathbf{X} . Note that $\mathbf{R}(v) = \mathbf{R}^T(-v)$ since $R_{12}(v) = R_{21}(-v)$ and R_{11} and R_{22} are symmetric about lag zero. Finally, we can write

$$\mathbf{R}(v) = \text{E} \left((\mathbf{X}(t) - \boldsymbol{\mu})(\mathbf{X}(t+v) - \boldsymbol{\mu})^T \right),$$

where $\boldsymbol{\mu} = \text{E}(X(t)) = (\mu_1, \mu_2)^T$.

The Cross-Correlation Function

If we define

$$\rho_{i,j}(v) = \text{Corr}(X_i(t), X_j(t+v)) = \frac{R_{ij}(v)}{\sqrt{R_{ii}(0)R_{jj}(0)}}, \quad v \in \mathcal{Z}, \quad i, j = 1, 2,$$

then ρ_{11} and ρ_{22} are the usual autocorrelation functions of X_1 and X_2 , while ρ_{12} is called the cross-correlation function of \mathbf{X} . If $\rho_{12}(v) = 0$ for all $v \in \mathcal{Z}$, then we say that the series X_1 and X_2 are not cross-correlated. We will often just say that they are uncorrelated if there is no chance of confusion over whether we mean un-autocorrelated or not cross-correlated.

Table 4.1. True Auto- and Cross-Correlations for a Bivariate Process

v	rho1(v)	rho2(v)	rho12(v)	rho12(-v)
0.	1.000	1.000	.038	.038
1.	.583	.517	.457	-.434
2.	.141	.068	.483	-.496
3.	-.136	-.183	.303	-.329
4.	-.220	-.236	.092	-.114
5.	-.174	-.168	-.050	.039
6.	-.081	-.067	-.102	.100
7.	-.003	.011	-.086	.091
8.	.038	.045	-.044	.049
9.	.043	.044	-.006	.009
10.	.028	.026	.016	-.015
11.	.010	.007	.020	-.021
12.	-.004	-.006	.014	-.015

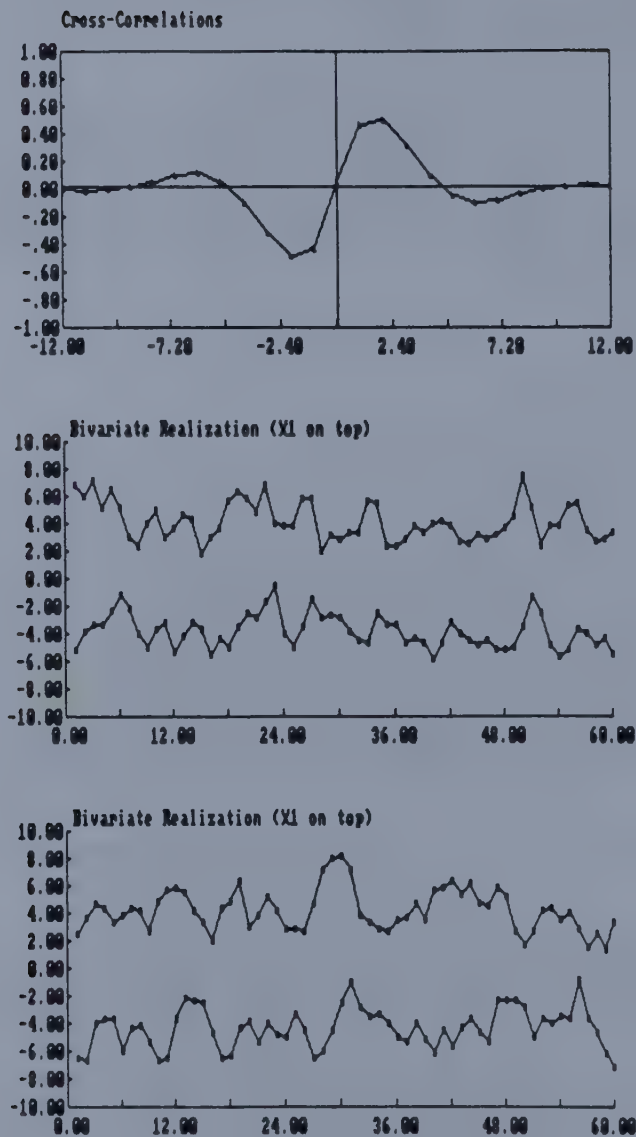


Figure 4.1. Plots of the Cross-Correlations Given in Table 4.1 and Two Realizations of Length 60 from the Process.

To illustrate the cross-correlation function of a bivariate time series, consider Table 4.1 where we have given the values of the true auto- and cross-correlations for a particular bivariate process. The cross-correlations are plotted in Figure 4.1 along with two realizations of length 60 from the process. In Example 4.1 we describe the macros that were used to produce this table and figure (see also Jenkins and Watts (1968), p. 332 for more information about this process). The cross-correlation of lag 0 is very small, indicating that the two series are practically uncorrelated simultaneously in time. The rest of the cross-correlations are approximately antisymmetric in nature; that is, they appear to be symmetric about zero except that those for negative lags are almost the negative of the ones for positive lags. This is an unusual phenomenon but affords an interesting example of trying to interpret what cross-correlations mean.

The fact that the correlations for lags 1, 2, and 3 are all large means that on the average the values of X_1 at time t and X_2 a little bit later (1, 2, or 3 time units) are similar in size. On the other hand, the correlations for lags -1 , -2 , and -3 are also large but negative; that is, on the average the values of X_1 at time t and X_2 a little bit earlier (again 1, 2, or 3 time units) are also similar in size, but on opposite sides of their respective means. In order to explain this, we consider the data plots. The univariate series tend to have what appear to be very crude "cycles." These are very irregular and are of varying lengths and shapes. We will see the spectral densities of X_1 and X_2 in Figure 4.2. These spectral densities have no sharp peaks but are consistent with the existence of the crude peaks and troughs in the data. If we look at the joint behavior of the cycles in the univariate series, we see that sometimes they are in phase and sometimes they are out of phase. In this context we consider two cycles to be "in phase" if peaks or troughs in the two series are almost aligned, while they are "out of phase" if a peak (or trough) in one series is almost aligned with a trough (or peak) in the other. The reason for the antisymmetric character of the cross-correlations is that when the cycles are in phase, X_1 tends to lead X_2 slightly; that is, the peak or trough in X_1 is followed by a similar peak or trough in X_2 a few time units later (sometimes 1 time unit, sometimes 2 or 3). On the other hand, when they are out of phase, the peak (or trough) in X_2 tends to occur slightly before the trough (or peak) in X_1 . We will discuss this example further in this chapter (see Problem C4.1 also). See Example 4.1 for the macros that were used to form Figure 4.1.

4.1.2. The Spectral Density Function

We next consider whether there exists a harmonic analysis of the autocovariance function R . We know from Section 2.1 that there are harmonic analyses for R_{11} and R_{22} . In our consideration of bivariate series we will assume that each of the three sequences R_{11} , R_{22} , and R_{12} is absolutely summable,

and write this as

$$\sum_{v=-\infty}^{\infty} |\mathbf{R}(v)| < \infty,$$

and say that \mathbf{R} is absolutely summable. Under this assumption we have the following theorem. Note that a matrix having complex elements is called a complex matrix, while the complex conjugate transpose of a complex matrix \mathbf{A} is denoted by \mathbf{A}^* and has (i, j) th element given by the complex conjugate of the (j, i) th element of \mathbf{A} .

Theorem 4.1.1 SPECTRAL REPRESENTATION OF \mathbf{R}

If the autocovariance function \mathbf{R} of a covariance stationary bivariate time series \mathbf{X} is absolutely summable, then

a) There exists a (2×2) complex matrix function \mathbf{f} , defined on $[0, 1]$, such that \mathbf{R} and \mathbf{f} are Fourier pairs; that is,

$$\mathbf{f}(\omega) = \sum_{v=-\infty}^{\infty} \mathbf{R}(v) e^{-2\pi i v \omega}, \quad \omega \in [0, 1]$$

$$\mathbf{R}(v) = \int_0^1 \mathbf{f}(\omega) e^{2\pi i v \omega}, \quad v \in \mathcal{Z}.$$

The sum and integral are calculated for each element of the matrices involved; for example, we have

$$f_{12}(\omega) = \sum_{v=-\infty}^{\infty} R_{12}(v) e^{-2\pi i v \omega}.$$

b) The matrix $\mathbf{f}(\omega)$ is Hermitian, that is, $\mathbf{f}^*(\omega) = \mathbf{f}(\omega)$, $\mathbf{f}(\omega)$ is positive semidefinite, that is,

$$\mathbf{h}^* \mathbf{f}(\omega) \mathbf{h} \geq 0$$

for any nonzero two-dimensional complex vector \mathbf{h} , and we have $\mathbf{f}(\omega) = \mathbf{f}^*(1 - \omega)$.

Proof: We prove part (b) only. Note that f_{11} and f_{22} are the usual univariate spectral densities for X_1 and X_2 and thus they are real and so are equal to their complex conjugates. We thus need only show that $\bar{f}_{12}(\omega) = f_{21}(\omega)$ in order to show that \mathbf{f} is Hermitian. We have

$$\bar{f}_{12}(\omega) = \sum_{v=-\infty}^{\infty} R_{12}(v) e^{2\pi i v \omega}$$

since (1) the complex conjugate of a sum is the sum of complex conjugates, (2) the complex conjugate of a real times a complex is the real times the complex conjugate of the complex, and (3) the complex conjugate of $e^{-2\pi i v \omega} = \cos 2\pi v \omega - i \sin 2\pi v \omega$ is $\cos 2\pi v \omega + i \sin 2\pi v \omega = e^{2\pi i v \omega}$. Now if we let $k = -v$, we get

$$\begin{aligned}\bar{f}_{12}(\omega) &= \sum_{k=-\infty}^{\infty} R_{12}(-k) e^{-2\pi i k \omega} \\ &= \sum_{k=-\infty}^{\infty} R_{21}(k) e^{-2\pi i k \omega} \\ &= f_{21}(\omega),\end{aligned}$$

since $R_{12}(-k) = R_{21}(k)$.

To show that $\mathbf{f}(\omega) = \mathbf{f}^*(1-\omega)$, we again need only show that $f_{12}(1-\omega) = \bar{f}_{21}(\omega)$ since f_{11} and f_{22} are symmetric about frequency .5. We have

$$\begin{aligned}f_{12}(1-\omega) &= \sum_{v=-\infty}^{\infty} R_{12}(v) e^{-2\pi i v(1-\omega)} \\ &= \sum_{v=-\infty}^{\infty} R_{12}(v) e^{2\pi i v \omega},\end{aligned}$$

since $e^{-2\pi i v} = 1$. But this last sum is clearly $\bar{f}_{21}(\omega)$.

We next show that $\mathbf{h}^T \mathbf{f}(\omega) \mathbf{h} \geq 0$ for all nonzero real two-dimensional vectors $\mathbf{h} = (h_1, h_2)^T$. If we let $Y(t) = h_1 X_1(t) + h_2 X_2(t)$, then it is easy to show (see Problem T4.1) that the univariate series Y has spectral density $\mathbf{h}^T \mathbf{f}(\omega) \mathbf{h}$. Recalling that the spectral density of a univariate time series is nonnegative gives the result. We can extend this to complex \mathbf{h} , but this requires the introduction of complex valued time series which we prefer not to do.

The spectral densities f_{11} and f_{22} are called autospectral densities, and the function f_{12} the cross-spectral density of \mathbf{X} . Before discussing the interpretation of the cross-spectral density function, we describe some functions that are derived from it.

Functions Derived from the Cross-Spectral Density

Since the cross-spectral density is complex valued, its value at a particular frequency contains two pieces of information, namely its real and imaginary parts. Thus we can write

$$f_{12}(\omega) = c_{12}(\omega) - i g_{12}(\omega),$$

where

$$c_{12}(\omega) = \operatorname{Re}(f_{12}(\omega)) \quad \text{and} \quad q_{12}(\omega) = -\operatorname{Im}(f_{12}(\omega))$$

are called the cospectral density and quadrature spectral density, respectively. We will see presently why q_{12} is defined to be the negative of the imaginary part of the cross-spectral density. We can also express the cross-spectral density in terms of its amplitude and phase (see Problem T1.2) by

$$f_{12}(\omega) = A_{12}(\omega)e^{i\phi_{12}(\omega)},$$

where

$$A_{12}(\omega) = |f_{12}(\omega)| \quad \text{and} \quad \phi_{12}(\omega) = \arctan \frac{-q_{12}(\omega)}{c_{12}(\omega)}$$

are called the amplitude spectrum and phase spectrum, respectively. See Problem T1.2 for a discussion of the arctangent function.

The POLAR Command

If the arrays **zr** and **zi** contain the real and imaginary parts a_j and b_j of the n complex numbers z_1, \dots, z_n , then the command

POLAR(zr,zi,n,amp,phase)

will return $A_j = |z_j|$ and $\phi_j = \arctan(b_j/a_j)$ in the arrays **amp** and **phase** of length **n**.

We can also define various standardized forms of the cross-spectral density, including the complex coherency function

$$w_{12}(\omega) = \frac{f_{12}(\omega)}{\sqrt{f_{11}(\omega)f_{22}(\omega)}},$$

the coherency function

$$W_{12}(\omega) = |w_{12}(\omega)|,$$

and the squared coherency function $W_{12}^2(\omega)$. Sometimes we will refer to the coherency, amplitude, and phase spectral densities as the coherence spectrum, amplitude spectrum, and phase spectrum, respectively. We will see that the squared coherency function plays the role of a multiple correlation coefficient. The next theorem shows that it lies between zero and one.

Theorem 4.1.2

RANGE OF COHERENCE SPECTRUM

If both autospectral densities are positive, then the coherency function and thus the squared coherency function are between zero and one for all frequencies.

Proof: The fact that $|w_{12}|$ is nonnegative follows from the fact that it is the ratio of nonnegative functions. The fact that $|w_{12}(\omega)| \leq 1$ follows from the fact that

$$f_{11}(\omega)f_{22}(\omega) - f_{12}(\omega)f_{21}(\omega) \geq 0,$$

which is true since (1) this is the determinant of $\mathbf{f}(\omega)$ which is positive semidefinite by part (b) of Theorem 4.1.1, and (2) the determinant of a positive semidefinite matrix is nonnegative.

Later we will estimate the functions defined above by substituting estimates of the auto- and cross-spectral densities into their formulas. Thus we will use only autospectral estimates that are guaranteed to be positive. Thus if one uses the truncated periodogram or Tukey windows to estimate the autospectra, the coherence need not be between zero and one. To illustrate the appearance of the squared coherence and phase spectra of a bivariate time series, consider Figure 4.2, which contains graphs of the autospectra, squared coherence, and phase spectra for the process in Figure 4.1. In Example 4.1 we describe how the auto- and cross-spectra were calculated for this series. The following macro can be used to form plots of the squared coherence and phase spectra for any process given (true or estimated) auto- and cross-spectra.

```

1 ;;
2 ;;   CROSSP.MAC: macro to find and plot the phase and squared
3 ;;               coherence functions given autospectra f11 and f22
4 ;;               and the real and imaginary parts f12r, f12i of the
5 ;;               cross-spectra of a time series.
6 ;;
7 ;;   INPUT: Q (the number of frequencies between 0 and 1)
8 ;;          f11, f22, f12r, f12i
9 ;;
10 PAUSE
11 ;start
12 q=Q/2
13 q=q+1
14 POLAR(f12r,f12i,q,amp,phase) ;POLAR gives arctan(b/a) not arctan(-b/a)
15 coher=amp*amp
16 coher=coher/f11
17 coher=coher/f22
18 LABEL(coher)='Squared Coherency Spectrum'
19 LABEL(phase)='Phase Spectrum'
20 aa=<0,1>

```

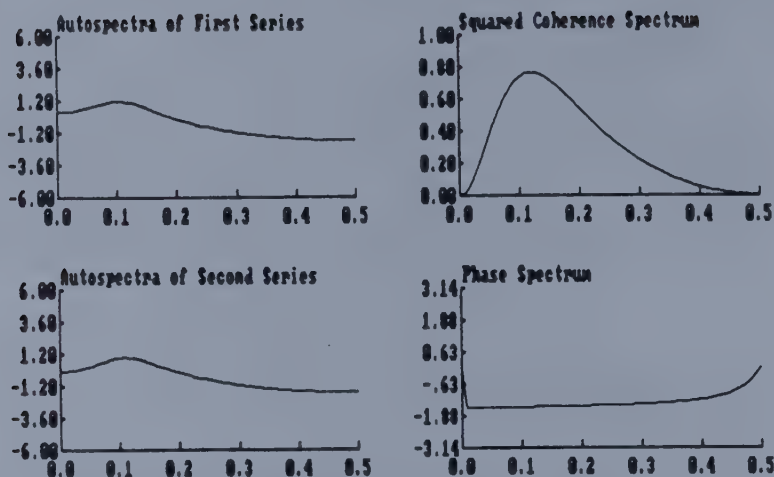


Figure 4.2. The Autospectra, Squared Coherence, and Phase Spectrum for the Process in Figure 4.1.

```

21 freq=POLY(aa,1,q,0,.5) ;find frequencies
22 LABEL(freq)= ' '
23 PLOT(freq,coher,q,0,.5,0,1)
24 mpi=-pi
25 PLOT(freq,phase,q,0,.5,mpi,pi)

```

Note that the autospectra in Figure 4.2 have some power in frequencies corresponding to the cycle lengths in the data sets in Figure 4.1. The phase spectrum is roughly constant and the squared coherence is large for moderate frequencies and small in its tails. We will discuss the interpretation of these functions next.

Interpreting Cross-Spectra

In Sections 2.1 and A.2, we interpreted the autospectral density at a frequency ω as the average value of the amplitudes of the frequency component of frequency ω for many realizations of a univariate time series. To interpret cross-spectra, we must visualize generating many long realizations of the bivariate series, and for each bivariate realization, calculating the coefficients of the cosine and sine terms in the sinusoidal decomposition of both univariate realizations. Thus for each bivariate realization, we will have four numbers for each frequency. Denote the cosine and sine coefficients for the frequency ω for a univariate realization by $a_i(\omega)$ and $b_i(\omega)$ where i is 1 for X_1 and 2 for X_2 . Then we have the following interpretation for the quantities derived from the

cross-spectral density.

Cospectrum and Quadrature Spectrum

The cospectrum $c_{12}(\omega)$ is the sum of the covariance of $a_1(\omega)$ with $a_2(\omega)$ and the covariance of $b_1(\omega)$ with $b_2(\omega)$. On the other hand, the quadrature spectrum $q_{12}(\omega)$ is the sum of the covariance of $a_1(\omega)$ with $b_2(\omega)$ and the covariance of $a_2(\omega)$ with $b_1(\omega)$. Note that the quadrature spectrum is defined to be the negative of the imaginary part of the cross-spectrum so that we do get this covariance interpretation. Thus the cospectrum for frequency ω measures the strength of the linear relationship between the coefficients over many realizations of like trigonometric functions (cosine or sine) for frequency ω , while the quadrature spectrum is for opposite trig functions. This is the origin of the prefix "co" and "quadrature."

Amplitude and Phase Spectra

The amplitude spectrum $A_{12}(\omega)$ combines the information about the coefficients contained in the co- and quadrature spectra. Thus it measures the relationship between the amplitudes of the sinusoids in the univariate realizations at frequency ω . The phase spectrum $\phi_{12}(\omega)$ measures how out of phase the frequency components for the univariate realizations tend to be. The phase is a number between $-\pi$ and π , with $-\pi$ or π indicating that the components are perfectly out of phase. For example, if $\phi_{12}(1/12) = \pi$, then the frequency components of period 12 tend to be perfectly out of phase; that is, if the data are monthly, then the 12-month frequency components are shifted 6 months from each other. If the autospectra indicate that the 12-month components are predominant, then $\phi_{12}(1/12) = \pi$ would indicate that the two series are rather cyclic (of period 12 months) but perfectly out of phase, with one series leading (or lagging) the other by 6 months. If, on the other hand, $\phi_{12}(1/12) = \pi/2$, then one series leads (or lags) the other by 3 months. We note that the phase does not indicate which series is leading the other.

The Coherence and Squared Coherence

The coherency spectrum is merely a normalized version of the amplitude spectrum, that is,

$$|w_{12}(\omega)| = \frac{A_{12}(\omega)}{\sqrt{f_{11}(\omega)f_{22}(\omega)}}.$$

If we recall the spectral representation of a univariate series, then we could show that the coherency spectrum at frequency ω is the absolute value of the correlation coefficient of the random amplitudes of the sinusoids of frequency ω in the two univariate series. Further, the squared coherence at frequency ω plays the role of the multiple correlation coefficient R^2 if we were to regress the

amplitudes of the frequency components of frequency ω in one series on the corresponding amplitudes for the other series over many bivariate realizations.

An Example of Interpreting Coherence

To illustrate the repeated realizations interpretation of the coherence, consider Figure 4.3 (see Example 4.2). In this figure, we generated 100 realizations each of length $n = 200$ from the bivariate time series that we discussed in Figures 4.1 and 4.2. For each bivariate realization we calculated the periodogram of the individual univariate realizations at $Q = 200$ frequencies between 0 and 1. We then extracted the pair of periodogram ordinates at frequencies .025, .125, and .40. Thus at the end of this experiment we had three sets of 100 pairs of amplitudes. The scatterplots of these three data sets are included in Figure 4.3. Note that the true coherence spectrum is large in the region around frequency .125 and small near frequencies 0 and .5. This is reflected in the three scatterplots. Those for frequencies .025 and .4 show no linear pattern, while the one for frequency .125 has evidence of strong correlation. Note that these scatterplots are somewhat unusual in appearance because they are for χ^2 variables since the periodogram has a χ^2 distribution. They seem unusual because we are used to looking at scatterplots of normally distributed random variables. Note also that the values of the autospectra (see Figure 4.2) are large in the region of frequency .125 and small near frequencies 0 and .5. This is also reflected in the scatterplots.

Bivariate White Noise

We next turn to the bivariate analog of a white noise time series. It is clear that we would like the univariate series X_1 and X_2 to be white noise, and that X_1 and X_2 should be uncorrelated at different times. The traditional definition of bivariate white noise processes allows there to be nonzero correlation between X_1 and X_2 at the same time point.

Definition. A bivariate time series \mathbf{X} is said to be a white noise series if $E(\mathbf{X}(t)) = \mathbf{0}$, while

$$R(v) = \begin{cases} \Sigma, & v = 0 \\ 0, & v \neq 0 \end{cases}$$

for some positive semidefinite matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}.$$

For such a process, we write $\mathbf{X} \sim \text{RW}(\Sigma)$.

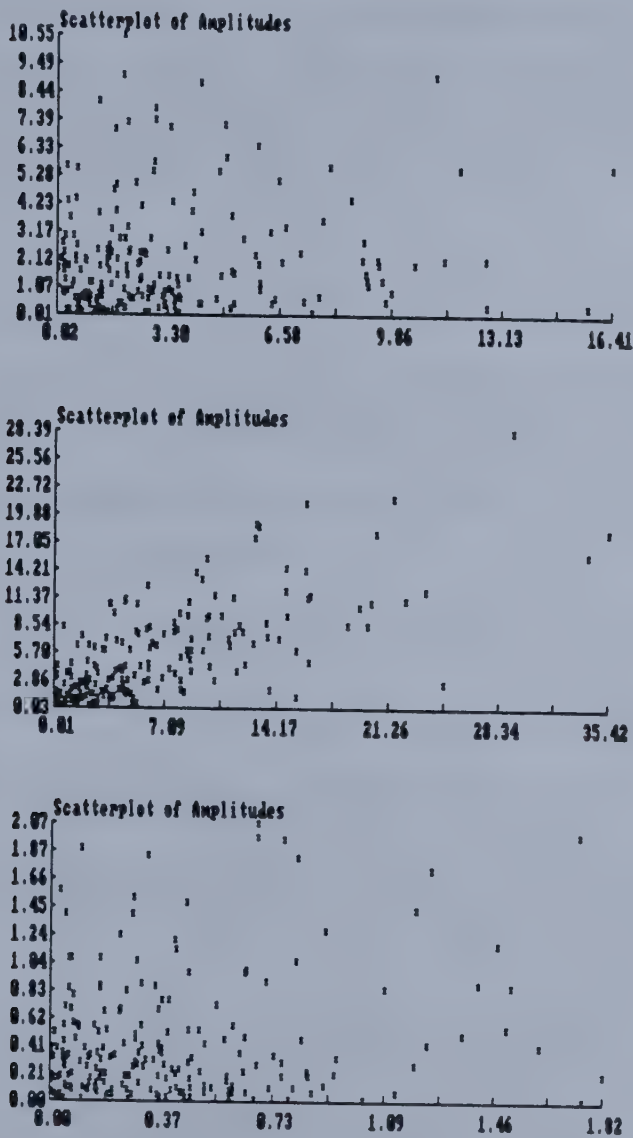


Figure 4.3. Interpreting the Coherence Spectrum of a Bivariate Time Series.

For a white noise series, we have

$$\rho_{12}(v) = \delta_v \frac{\sigma_{12}}{\sigma_1 \sigma_2}$$

$$f_{12}(\omega) = \sigma_{12}$$

$$|w_{12}(\omega)| = \frac{|\sigma_{12}|}{\sigma_1 \sigma_2}$$

$$\phi_{12}(\omega) = 0.$$

Thus the cross-spectral density is also a constant, while the coherence is a constant equal to the simultaneous correlation between X_1 and X_2 and the phase spectrum is identically zero. Thus on the average, the frequency components in the two series are in phase.

Two other important uses of the coherency spectrum are described in the next theorem.

Theorem 4.1.3

TWO PROPERTIES OF COHERENCE

Let \mathbf{X} be a covariance stationary bivariate time series with autocovariance function \mathbf{R} and spectral density function \mathbf{f} . Then

a) The univariate series X_1 and X_2 are not cross-correlated if and only if the coherency spectrum is zero for all frequencies.

b) If Y_1 and Y_2 are filtered versions of X_1 and X_2 , then the coherency spectrum of \mathbf{Y} is the same as that of \mathbf{X} ; that is, coherency is unaffected by linear transformations.

Proof: Part (a) follows immediately from the fact that

$$R_{12}(v) = \int_0^1 f_{12}(\omega) e^{2\pi i v \omega} d\omega,$$

and the Fourier coefficients of a function are all zero if and only if the function is identically zero. In Problem T4.2 we discuss the proof of part (b), which follows from the bivariate Filter Theorem which we consider next.

4.1.3. Bivariate Filters

In Section 2.3 we discussed determining the properties of the output of a filter from that of the input. We now extend these results to include the calculation of the cross-covariances and cross-spectra between the input and the output. We also consider the more general case where a bivariate series \mathbf{Y} is a filtered version of the bivariate series \mathbf{X} .

Definition. The bivariate time series \mathbf{Y} is said to be a filtered version of the bivariate series \mathbf{X} with filter coefficient matrices $\{\mathbf{B}(j), j \in \mathcal{Z}\}$ and filter operator $\mathbf{G}(L) = \sum_{j=-\infty}^{\infty} \mathbf{B}(j)L^j$ if

$$\mathbf{Y}(t) = \sum_{j=-\infty}^{\infty} \mathbf{B}(j)\mathbf{X}(t-j) = \mathbf{G}(L)\mathbf{X}(t)$$

exists as a limit in mean square.

An example of this bivariate filter is the bivariate moving average process \mathbf{X} defined by

$$\mathbf{X}(t) = \sum_{k=0}^q \mathbf{B}(k)\boldsymbol{\epsilon}(t-k),$$

where $\boldsymbol{\epsilon}$ is a bivariate white noise series.

Theorem 4.1.4	BIVARIATE FILTER THEOREM
----------------------	---------------------------------

a) If the univariate time series X_2 is a filtered version of the univariate series X_1 , that is,

$$X_2(t) = \sum_{j=-\infty}^{\infty} \beta_j X_1(t-j),$$

then

i) The cross-covariance function R_{12} satisfies

$$R_{12}(v) = \sum_{j=-\infty}^{\infty} \beta_j R_{11}(v-j), \quad v \in \mathcal{Z}.$$

ii) The cross-spectral density function f_{12} is given by

$$f_{12}(\omega) = g(e^{-2\pi i \omega}) f_{11}(\omega),$$

where $g(z) = \sum_{j=-\infty}^{\infty} \beta_j z^j$.

b) If the bivariate series \mathbf{Y} is a filtered version of the covariance stationary bivariate series \mathbf{X} , then \mathbf{Y} is also covariance stationary and the autocovariance and spectral density functions \mathbf{R}_Y and \mathbf{f}_Y of \mathbf{Y} are related to those of \mathbf{X} by

$$\mathbf{R}_Y(v) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \mathbf{B}(j) \mathbf{R}_X(j+v-k) \mathbf{B}^T(k)$$

$$\mathbf{f}_Y(\omega) = \mathbf{G}(e^{2\pi i \omega}) \mathbf{f}_X(\omega) \mathbf{G}^*(e^{2\pi i \omega}).$$

Proof of part (b): For convenience we assume that the mean vector for \mathbf{X} is the vector of zeros. Thus we have

$$\begin{aligned} \mathbf{R}_Y(v) &= \mathbf{E}(\mathbf{X}(t) \mathbf{X}^T(t+v)) \\ &= \mathbf{E} \sum_{j=-\infty}^{\infty} \mathbf{B}(j) \mathbf{X}(t-j) \left(\sum_{k=-\infty}^{\infty} \mathbf{B}(k) \mathbf{X}(t+v-k) \right)^T \\ &= \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \mathbf{B}(j) \mathbf{E}(\mathbf{X}(t-j) \mathbf{X}^T(t+v-k)) \mathbf{B}^T(k) \\ &= \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \mathbf{B}(j) \mathbf{R}_X(j+v-k) \mathbf{B}^T(k). \end{aligned}$$

From this expression one can use some tedious algebra to verify the expression for \mathbf{f}_Y .

We note the similarity of the expressions in this theorem to those in the univariate case (see Problem T4.3). The difficulty in the bivariate case is that the matrices involved do not commute. Thus \mathbf{f}_Y is the product of three matrices and we cannot alter the order in which they are multiplied.

4.1.4. Filters and Cross-Spectral Analysis

The use of the cross-spectral density function to study a bivariate time series is called cross-spectral analysis. In this section we consider using cross-spectral analysis in two ways. First we use it to describe the effects of a filter being applied to a univariate series. Then we will see how cross-spectral analysis can be employed to determine whether one univariate series can be

thought of as a filtered version of another. This is the bivariate time series analog of simple linear regression. Note that we will be assuming that we know the quantities involved. In Section 4.2 we discuss using these methods when we must estimate the cross-spectra.

Describing the Effects of a Filter

Suppose that the univariate series X_2 is a filtered version of the univariate series X_1 ; that is, suppose that

$$X_2(t) = \sum_{j=-\infty}^{\infty} \beta_j X_1(t-j) = g(L)X_1(t).$$

Then we have by the bivariate Filter Theorem that

$$f_{12}(\omega) = g(e^{-2\pi i\omega})f_{11}(\omega) \quad \text{and} \quad f_{22}(\omega) = |g(e^{2\pi i\omega})|^2 f_{11}(\omega),$$

which means that the coherence spectrum of X_1 and X_2 is given by

$$\begin{aligned} W_{12}(\omega) &= \frac{|f_{12}(\omega)|}{\sqrt{f_{11}(\omega)f_{22}(\omega)}} \\ &= \frac{|g(e^{2\pi i\omega})|f_{11}(\omega)}{\sqrt{f_{11}(\omega)|g(e^{2\pi i\omega})|^2 f_{11}(\omega)}} \\ &= 1, \quad \omega \in [0, 1], \end{aligned}$$

which means that the amplitudes of the frequency components for a specified frequency in the input and output series are perfectly correlated, and this is true for every frequency (see Problem C4.2). Now two random variables (such as these two amplitudes) are perfectly correlated if and only if one is a linear function of the other. To get an idea of what the filter does to the amplitude of the frequency component of frequency ω in the input series, we recall that the spectral representation of a univariate time series allows us to think of such a series as being made up of the sum of many sinusoids and that these sinusoids can be considered separately. Thus we find what the filter would do to a series consisting of just a sinusoid of frequency ω , that is, to $X_1(t) = \cos 2\pi t\omega$. We

have

$$\begin{aligned}
 X_2(t) &= \sum_{j=-\infty}^{\infty} \beta_j X_1(t-j) = \sum_{j=-\infty}^{\infty} \beta_j \cos 2\pi(t-j)\omega \\
 &= \sum_{j=-\infty}^{\infty} \beta_j [\cos 2\pi t\omega \cos 2\pi j\omega + \sin 2\pi t\omega \sin 2\pi j\omega] \\
 &= \cos 2\pi t\omega \left[\sum_{j=-\infty}^{\infty} \beta_j \cos 2\pi j\omega \right] + \sin 2\pi t\omega \left[\sum_{j=-\infty}^{\infty} \beta_j \sin 2\pi j\omega \right] \\
 &= \cos 2\pi t\omega \operatorname{Re} [g(e^{2\pi i\omega})] + \sin 2\pi t\omega \operatorname{Im} [g(e^{2\pi i\omega})] \\
 &= |g(e^{2\pi i\omega})| \cos (2\pi t\omega + \phi(\omega)),
 \end{aligned}$$

where

$$\phi(\omega) = \arctan \frac{\operatorname{Re} (g(e^{2\pi i\omega}))}{\operatorname{Im} (g(e^{2\pi i\omega}))}.$$

Thus the output of the filter is also a sinusoid of frequency ω , except (1) it has been multiplied by the modulus of the frequency transfer function of the filter (which is also known as the gain of the filter), and (2) it has been shifted out of phase with the input sinusoid by the amount determined by ϕ (see Problem T1.3 for a review of phase shifts). In analogy with terminology in electronics, we call the function

$$g_{2|1}(\omega) = \frac{|f_{12}(\omega)|}{f_{11}(\omega)}$$

the gain spectrum of X_2 given X_1 . Thus if X_2 is in fact a filtered version of X_1 , then the gain and phase spectra will be the same as the gain and phase of the filter.

The Phase Spectrum of a Delay Process

One important relationship between univariate series X_1 and X_2 is the case where

$$X_2(t) = \beta X_1(t-d) + \epsilon(t)$$

for some lag d , where ϵ is a white noise process having variance σ^2 and is uncorrelated with X_1 . Thus $X_2(t)$ is a noisy and time delayed version of X_1 . For convenience we assume that X_1 has zero mean. Thus we have

$$\begin{aligned}
 R_{12}(v) &= E[X_1(t)(X_1(t+v-d) + \epsilon(t+v))] \\
 &= R_{11}(v-d),
 \end{aligned}$$

and thus

$$\begin{aligned}
 f_{12}(\omega) &= \sum_{v=-\infty}^{\infty} R_{11}(v-d)e^{-2\pi i v \omega} \\
 &= e^{-2\pi i d \omega} \sum_{v=-\infty}^{\infty} R_{11}(v-d)e^{-2\pi i (v-d)\omega} \\
 &= e^{-2\pi i d \omega} f_{11}(\omega) \\
 &= \cos 2\pi d \omega f_{11}(\omega) - i \sin 2\pi d \omega f_{11}(\omega).
 \end{aligned}$$

This means that the phase spectrum is

$$\begin{aligned}
 \phi_{12}(\omega) &= \arctan \frac{-\sin 2\pi d \omega}{\cos 2\pi d \omega} \\
 &= \arctan(-\tan 2\pi d \omega) \\
 &= -2\pi d \omega;
 \end{aligned}$$

that is, the phase spectrum is a linear function of ω with slope given by $2\pi d$.

Determining an Optimal Filter

Given two univariate series X_1 and X_2 , we now seek to find the filter $\{\beta_j, j \in \mathcal{Z}\}$ such that

$$E \left[X_2(t) - \sum_{j=-\infty}^{\infty} \beta_j X_1(t-j) \right]^2$$

is as small as possible; that is, we seek the filter that when applied to X_1 results in the series that is as close to X_2 (in the mean square error sense) as possible. The solution to this problem is provided in the next theorem.

Theorem 4.1.5 OPTIMAL FILTERS

Let $\mathbf{X}(t) = (X_1(t), X_2(t))^T$ be a covariance stationary bivariate time series having spectral density \mathbf{f} . Then

a) The filter that when applied to X_1 best approximates X_2 in the mean square error sense is given by

$$\beta_j = \int_0^1 \frac{f_{12}(\omega)}{f_{11}(\omega)} e^{2\pi i j \omega} d\omega, \quad j \in \mathcal{Z}.$$

b) *The residual time series*

$$\epsilon(t) = X_2(t) - \sum_{j=-\infty}^{\infty} \beta_j X_1(t-j)$$

has spectral density function

$$\begin{aligned} f_{\epsilon}(\omega) &= f_{22}(\omega) - f_{21}(\omega) f_{11}^{-1}(\omega) f_{12}(\omega) \\ &= f_{22}(\omega) - \frac{|f_{12}(\omega)|^2}{f_{11}(\omega)} \\ &= f_{22}(\omega) [1 - W_{12}^2(\omega)], \end{aligned}$$

which means that the filtered version of X_1 perfectly matches X_2 for the frequency component of frequency ω if and only if the squared coherence W_{12}^2 is one at frequency ω .

Implications: This theorem essentially provides a regression analysis of one univariate series on another, except that there is a regression for each frequency, with the squared coherence playing the role of R^2 at each frequency. Note further the similarity of the form of the functions involved to those in ordinary regression analysis if we think of \mathbf{X} as the “input” to the regression model and \mathbf{y} as the “output.” For example, the filter coefficients are given as the Fourier transform of the function f_{12}/f_{11} (which is called the regression transfer function), while least squares regression coefficients are given by $\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$, which is of a similar form with $\mathbf{X}^T \mathbf{X}$ and $\mathbf{X}^T \mathbf{y}$ playing the roles of f_{11} and f_{12} . We also have that the residual sum of squares in regression is of the form $\text{RSS} = \mathbf{y}^T \mathbf{y} - \mathbf{y}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{y}$, which is analogous to the formula for the spectral density of the error time series.

Calculating Filter Coefficients

In practice, we will only know the values of f_{12} and f_{11} (or estimates of them) at the frequencies $\omega_j = (j-1)/Q$, $j = 1, \dots, Q$, for some integer Q . Thus we will approximate the integral in part (a) of Theorem 4.1.5 by the rectangular sum

$$\tilde{\beta}_j = \frac{1}{Q} \sum_{k=1}^Q \frac{f_{12}(\omega_k)}{f_{11}(\omega_k)} e^{2\pi i j \omega_k}.$$

We have the following results for this approximation.

Theorem 4.1.6

APPROXIMATING FILTER COEFFICIENTS

Let $\tilde{\beta}_j$ be the rectangular sum approximation (based on Q frequencies) to the filter coefficient β_j . Then

a) For $j \geq 0$, $\tilde{\beta}_j$ is the real part of the $(j+1)$ st element of the discrete Fourier transform of

$$\frac{f_{12}(\omega_1)}{f_{11}(\omega_1)}, \dots, \frac{f_{12}(\omega_Q)}{f_{11}(\omega_Q)}.$$

b) $\tilde{\beta}_{-j} = \tilde{\beta}_{Q-j}$.

c) If $\beta_j = 0$ for $|j| \geq Q$, then $\tilde{\beta}_j = \beta_j$.

Thus to find the coefficients of an optimal filter, we will (1) evaluate (or estimate) the auto- and cross-spectra, and then (2) use the FFT command to approximate the integrals involved. See Problem T4.6 for a further simplification of part (a).

The Double Command

As in the univariate case, we will usually calculate spectral functions at the frequencies $(j-1)/Q$ for $j = 1, \dots, Q$, but store only the first $q = [Q/2] + 1$ values. To use the FFT command to do the rectangular sums approximation, we will need the spectral arrays at all Q frequencies. Thus given the real and imaginary parts of f_{12} at $q = [Q/2] + 1$ frequencies in the arrays **f12r** and **f12i**, the command

DOUBLE(f12r,f12i,Q)

extends the arrays **f12r** and **f12i** to length Q by using the facts that the real part is symmetric about 0.5, while the imaginary part is antisymmetric about 0.5; that is, the values at frequencies ω and $1 - \omega$ are the same except they have opposite signs.

The following macro will calculate the $\tilde{\beta}_j$'s corresponding to arrays **f12r**, **f12i**, and **f11**.

```

1 ;;
2 ;;   FINDFILT.MAC: This macro accepts the cross-spectra
3 ;;           fxyr and fxyi of two series and the
4 ;;           autospectra of the first in fxx, and then
5 ;;           approximates the integrals in the Fourier
6 ;;           transform to find the best fitting filter
7 ;;           coefficients.
8 ;;
9 ;;   INPUT: fxyr, fxyi, fxx (cross-spectra and autospectra at
```

```

10 ;;          q=[Q/2]+1 points)
11 ;;          Q (number of frequencies between 0 and 1 at which spectra
12 ;;          were calculated).
13 ;;          m (number of coefficients to find)
14 ;;
15 PAUSE
16 ;start
17 fxyr1=fxyr/fxx          ;divide by fxx
18 fxyi1=fxyi/fxx
19 DOUBLE(fxyr1,fxyi1,Q)   ;double arrays by symmetry
20 mp1=m+1
21 FFT(fxyr1,fxyi1,Q,mp1,1,fxyr1,fxyi1)
22 beta0=fxyr1[1]/Q        ;pick off first element
23 beta=EXTRACT(fxyr1,2,mp1) ;get the next m
24 beta=beta/Q
25 LABEL(beta)='Filter Coefficients'
26 LIST(beta0)
27 LIST(beta)

```

As a simple example, suppose

$$X_2(t) = \frac{1}{2} \sum_{j=0}^{\infty} \left(\frac{1}{2}\right)^j X_1(t-j),$$

where X_1 is white noise with variance 1. Thus (see Problem T4.7)

$$f_{12}(\omega) = \frac{\frac{1}{2} \left(1 - \frac{1}{2} e^{2\pi i \omega}\right)}{\left|1 - \frac{1}{2} e^{-2\pi i \omega}\right|^2}.$$

The next macro calculates this function at a user-specified number of frequencies and then calls the FINDFIL macro to approximate the filter coefficients.

```

1 ;;
2 ;;   FILTEX.MAC: This macro finds the transfer function of the
3 ;;   filter
4 ;;
5 ;;           y(t)= 1/2 [ Sum(j=0,infinity) (1/2)^j X(t-j)]
6 ;;
7 ;;           and then calls FINDFIL to find the coefficients.
8 ;;
9 ;;   INPUT: Q (number of frequencies between 0 and 1)
10 ;;          m (number of coefficients to find)
11 ;;
12 PAUSE
13 ;start
14 q=[Q/2]+1
15 freq=LINE(q,0,.5,1) ;freq contains [Q/2]+1 frequencies from 0 to .5
16 freq=2*pi*freq
17 cc=COS(freq,q)

```

```

18 ss=SIN(freq,q)
19 fxyr=1-.5*cc
20 fxyi={-.5}*ss
21 d=2*(fxyr^2+fxyi^2)
22 fxyr=fxyr/d
23 fxyi=fxyi/d
24 fxx=LINE(q,1,0)
25 fyy=LINE(q,1,0)
26 MACRO(findfilt,start)

```

If we use this macro for $Q = 10$, then the β 's are determined correctly to five decimal places. Note that the amplitude of this transfer function is very smooth and thus the approximation is quite good even for small Q . In cases where the cross-spectrum is not so smooth, we will have to use larger values of Q .

4.1.5. The Bivariate Autoregressive Process

As in the univariate case, the bivariate autoregressive process is very important in modeling data. A time series \mathbf{X} is said to be an autoregressive process of order p with (2×2) coefficient matrices $\mathbf{A}(1), \dots, \mathbf{A}(p)$ and error covariance matrix Σ if the process satisfies the stochastic difference equation

$$\sum_{j=0}^p \mathbf{A}(j) \mathbf{X}(t-j) = \boldsymbol{\epsilon}(t), \quad t \in \mathcal{Z},$$

where $\boldsymbol{\epsilon}$ is a bivariate white noise time series with covariance matrix Σ . Note that this equality must be interpreted in a fashion similar to the interpretation for the univariate case. Thus we define the autoregressive operator

$$\mathbf{G}(\mathbf{L}) = \sum_{j=0}^p \mathbf{A}(j) \mathbf{L}^j.$$

If we can invert \mathbf{G} in terms of nonnegative powers of \mathbf{L} only, that is, as

$$\mathbf{G}^{-1}(\mathbf{L}) = \sum_{k=0}^{\infty} \mathbf{B}(k) \mathbf{L}^k,$$

then we have that

$$\mathbf{X}(t) = \sum_{k=0}^{\infty} \mathbf{B}(k) \boldsymbol{\epsilon}(t-k)$$

exists as a limit in mean square and satisfies the AR difference equation. To obtain the \mathbf{B} 's, we equate like powers of z in the equation $\mathbf{G}(\mathbf{L})\mathbf{G}^{-1}(\mathbf{L}) = \mathbf{I}_2$ and obtain $\mathbf{B}(0) = \mathbf{I}_2$ and

$$\mathbf{B}(j) = - \sum_{i=1}^{\min(p,j)} \mathbf{A}(i) \mathbf{B}(j-i), \quad j \geq 1.$$

In the univariate case, the AR operator can be inverted in terms of nonnegative powers if and only if the zeros of the associated complex polynomial $g(z) = \sum_{j=0}^p \alpha_j z^j$ are all outside the unit circle. The analogous condition in the bivariate case is that the zeros of the determinant of the matrix of polynomials $G(z) = \sum_{j=0}^p \mathbf{A}(j) z^j$ are all outside the unit circle. We state this as a theorem.

Theorem 4.1.7 STATIONARITY OF AR

The stochastic difference equation defining a bivariate AR process has a stationary solution in terms of present and past ϵ 's if and only if the zeros of

$$\left| \sum_{j=0}^p \mathbf{A}(j) z^j \right|$$

are all outside the unit circle.

To illustrate this theorem, consider $p = 1$ and denote the single coefficient matrix by

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

Then we have

$$\begin{aligned} \left| \sum_{j=0}^p \mathbf{A}(j) z^j \right| &= |\mathbf{I}_2 + \mathbf{A}z| \\ &= \begin{vmatrix} 1 + a_{11}z & a_{12}z \\ a_{21}z & 1 + a_{22}z \end{vmatrix} \\ &= 1 + (a_{11} + a_{22})z + (a_{11}a_{22} - a_{12}a_{21}) \\ &= 1 + \text{tr}(\mathbf{A})z + |\mathbf{A}|z^2. \end{aligned}$$

Note that for general p , the determinantal polynomial will be of degree $2p$ and the coefficients will not be so easily determined as in this example (see Problem T4.8). In the first-order case, we can operate as though the second-degree polynomial is that of a univariate second-order process with coefficients

$$\alpha_1 = \text{tr}(\mathbf{A}) \quad \text{and} \quad \alpha_2 = |\mathbf{A}|.$$

This univariate process has partial autocorrelations (see Theorem 2.6.4) $\theta_2 = -\alpha_2$ and $\theta_1 = -\alpha_1/(1 + \alpha_2)$. Thus the bivariate process is stationary if and

only if (1) the absolute value of the determinant of \mathbf{A} is less than 1, and (2) the ratio of the trace of \mathbf{A} to 1 plus its determinant is less than 1 in absolute value.

The bivariate time series that we described in Section 4.1 (see Figures 4.1 and 4.2) is the autoregressive process having coefficient matrix \mathbf{A} and error covariance matrix Σ given by

$$\mathbf{A} = \begin{bmatrix} -0.6 & 0.5 \\ -0.4 & -0.5 \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}.$$

Thus if we let ϵ_1 and ϵ_2 be two independent univariate Gaussian white noise time series, each having variance 1, we have

$$X_1(t) = 0.6X_1(t-1) - 0.5X_2(t-1) + \epsilon_1(t)$$

$$X_2(t) = 0.4X_1(t-1) + 0.5X_2(t-1) + \epsilon_2(t).$$

Note that the coefficients have changed signs from those in the matrix \mathbf{A} because we have put $X_1(t-1)$ and $X_2(t-1)$ on the right side of the equal sign. This model says that X_1 at time t is approximately the difference of X_1 and X_2 at the previous time (plus noise), while X_2 is the almost the average of X_1 and X_2 at the previous time (also plus noise). The cross-correlation and cross-spectral density functions summarize the joint behavior of the two series.

The ARCORR2 Command

If the zeros of $|\mathbf{G}(z)|$ are all outside of the unit circle, then we have

$$\mathbf{X}(t) = \sum_{k=0}^{\infty} \mathbf{B}(k)\epsilon(t-k),$$

which means by the bivariate Filter Theorem that

$$\mathbf{R}(v) = \text{Cov}(\mathbf{X}(t), \mathbf{X}(t+v)) = \sum_{k=0}^{\infty} \mathbf{B}(k)\Sigma\mathbf{B}^T(k+v).$$

The ARCORR2 command uses this series to find the autocovariances. If it does not converge within a reasonable time (ARCORR2 will use as many as 100 terms in the sum), then ARCORR2 issues a message that the process is not stationary and aborts. The command is of the form

ARCORR2(A,p,SIGMA,M,R10,R20,rho120,rho1,rho2,rho12,rho21,ier)

where \mathbf{M} is the number of lags at which correlations are wanted. The output consists of the variances R10 and R20 of the univariate series, the cross-correlation

of lag zero in **rho120**, the first **M** autocorrelations of the univariate series in the arrays **rho1** and **rho2**, and finally the cross-correlations $\rho_{12}(k)$ for $k = 1, \dots, M$ in the array **rho12** and $\rho_{21}(k)$ for $k = 1, \dots, M$ in the array **rho21**. Recall that $\rho_{21}(v) = \rho_{12}(-v)$, and thus **rho12** (**rho21**) contains cross-correlations between $X_1(t)$ and X_2 at later (earlier) times. If the process is judged to be nonstationary, the output integer variable **ier** is set to 1 and no other output is returned. If the process is judged to be stationary, **ier** is set to 0. See Example 4.1 for an illustration of the use of the **ARCORR2** command.

The ARDT2 Command

As in the univariate case (see the **ARDT** command), **TIMESLAB** has two forms of the command for simulating bivariate AR processes:

X=ARDT2(A,p,n,e,ier)

and

X=ARDT2(A,p,sigma,seed,n,ier).

The first form actually finds the future terms of a general bivariate difference equation

$$\mathbf{X}(t) = \mathbf{e}(t) - \sum_{j=1}^p \mathbf{A}(j)\mathbf{X}(t-j), \quad t = p+1, \dots, n,$$

where for $t = 1, \dots, p$, $\mathbf{X}(t)$ is set equal to the starting values $\mathbf{e}(t)$. The **A**'s and **e**'s are entered in the arrays **A** and **e**, while the order p and length n are entered in the integers **p** and **n**. The output integer **ier** is 1 if any term in the difference equation becomes larger than 10^{10} . As in the univariate case, if the zeros of the determinantal polynomial are not outside the unit circle, the terms of the sum can become explosive, and thus **TIMESLAB** guards against this possibility. If it happens, then no **X** is returned. If the terms are not explosive, then **ier** is set equal to 0. When using this command it is important to remember that matrices (such as **e** and **X**) are stored in **TIMESLAB** by column, while three-dimensional arrays (such as **A**) are stored first by column and then by index. For example if we have a second-order process with

$$\mathbf{A}(1) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \text{and} \quad \mathbf{A}(2) = \begin{bmatrix} e & f \\ g & h \end{bmatrix},$$

then the array **A** should be of the form

A=<a,c,b,d,e,g,f,h>.

The second form of the **ARDT2** command generates a realization having Gaussian errors, and has the additional input arguments **sigma** and **seed** containing the desired error covariance matrix and random number generator seed. The output integer **ier** has the same meaning as in the first form of the command. Note that the second form of the **ARDT2** command was used to generate the data in Figure 4.1 (see Example 4.1).

The ARSP2 Command

From the bivariate Filter Theorem we have that the spectral density function of the white noise error series is given by

$$\mathbf{f}_\epsilon(\omega) = \Sigma = \mathbf{G}(e^{2\pi i\omega})\mathbf{f}_X(\omega)\mathbf{G}^*(e^{2\pi i\omega}),$$

and thus since the determinant of $\mathbf{G}(e^{2\pi i\omega})$ is never zero for $\omega \in [0, 1]$, we can invert it (and its complex conjugate transpose) to obtain

$$\mathbf{f}_X(\omega) = \mathbf{G}^{-1}(e^{2\pi i\omega})\Sigma\mathbf{G}^{-*}(e^{2\pi i\omega}),$$

where by \mathbf{G}^{-*} we mean the inverse of the complex conjugate of the matrix \mathbf{G} . The command

ARSP2(A,p,sigma,q,f11,f22,f12r,f12i)

returns the autospectra f_{11} and f_{22} in the arrays **f11** and **f22** and the real and imaginary parts of the cross-spectra f_{12} in the arrays **f12r** and **f12i**, all at the frequencies $(j-1)/q$ for $j = 1, \dots, [q/2] + 1$, where q is entered in the integer **q**. The FFT algorithm is not used in this command, so **q** need not satisfy any of the rules for number of frequencies as in the univariate case. Again see Example 4.1 for an illustration of the use of **ARSP2**.

The Bivariate Yule-Walker Equations and Levinson Algorithm

We next turn to the bivariate analog of the Yule-Walker equations. If we multiply the bivariate AR stochastic difference equation on the right by $\mathbf{X}^T(t-v)$ and take expected values of both sides of the result, we obtain

$$\sum_{j=0}^p \mathbf{A}(j)\mathbf{R}(j-v) = \delta_v \Sigma, \quad v \geq 0,$$

where δ_v is the Kronecker delta function; that is, δ_v is zero unless $v = 0$ in which case it is one. This is again because $\mathbf{X}(t-v)$ is uncorrelated with $\epsilon(t)$ for positive v , while

$$\text{Cov}(\mathbf{X}(t), \epsilon(t)) = \Sigma.$$

Thus we again have a system of linear equations (called the bivariate Yule-Walker equations) relating the coefficients and autocovariances of a bivariate AR process. If we know the autocovariances $\mathbf{R}(0), \mathbf{R}(1), \dots, \mathbf{R}(p)$, we can find the corresponding coefficients and error covariance matrix by the bivariate Levinson algorithm given in the next theorem. This algorithm is due to Whittle (1963b).

Theorem 4.1.8 BIVARIATE LEVINSON ALGORITHM

Suppose that \mathbf{X} is a bivariate AR process of order p with coefficients $\mathbf{A}(1), \dots, \mathbf{A}(p)$ and noise variance Σ . For $k = 1, \dots, p$, let the matrices $\mathbf{A}_k(j)$, $\mathbf{B}_k(j)$, $j = 1, \dots, k$, and Σ_k and Γ_k be the solutions to the systems of equations

$$\sum_{j=0}^k \mathbf{A}_k(j) \mathbf{R}(v-j) = \delta_k \Sigma_k, \quad \sum_{j=0}^k \mathbf{B}_k(j) \mathbf{R}(j-v) = \delta_k \Gamma_k, \quad v = 0, \dots, k.$$

Then

a) These matrices can be found recursively by defining $\Sigma_0 = \Gamma_0 = \mathbf{R}(0)$ and $\Delta_0 = \mathbf{R}^T(1)$, $\Phi_0 = \mathbf{R}(1)$, and then for $k = 1, \dots, p$:

$$\mathbf{A}_k(k) = -\Delta_{k-1} \Gamma_{k-1}^{-1}, \quad \mathbf{B}_k(k) = -\Phi_{k-1} \Sigma_{k-1}^{-1},$$

$$\mathbf{A}_k(j) = \mathbf{A}_{k-1}(j) + \mathbf{A}_k(k) \mathbf{B}_{k-1}(k-j), \quad j = 1, \dots, k-1$$

$$\mathbf{B}_k(j) = \mathbf{B}_{k-1}(j) + \mathbf{B}_k(k) \mathbf{A}_{k-1}(k-j), \quad j = 1, \dots, k-1$$

$$\Sigma_k = \sum_{j=0}^k \mathbf{A}_k(j) \mathbf{R}(j), \quad \Gamma_k = \sum_{j=0}^k \mathbf{B}_k(j) \mathbf{R}^T(j)$$

$$\Delta_k = \sum_{j=0}^k \mathbf{A}_k(j) \mathbf{R}^T(k+1-j), \quad \Phi_k = \sum_{j=0}^k \mathbf{B}_k(j) \mathbf{R}(k+1-j).$$

b) The coefficients of the AR process are given by $\mathbf{A}_p(j)$, $j = 0, \dots, p$, while $\Sigma = \Sigma_p$.

c) The coefficients and prediction error covariance matrix of the best linear unbiased predictor of $\mathbf{X}(t)$ given the previous k \mathbf{X} 's are given by the \mathbf{A}_k 's and Γ_k , while those for $\mathbf{X}(t)$ given the next k \mathbf{X} 's are given by the \mathbf{B}_k 's and Φ_k .

In the univariate case, the forward and backward prediction coefficients in Levinson's algorithm were the same (except in reverse order) and the algorithm

greatly simplified. Note that any coefficient matrix having index zero is the identity matrix. Also, when $k = 1$, we skip the steps for $\mathbf{A}_k(j)$ and $\mathbf{B}_k(j)$, while when $k = p$, we don't calculate Δ_k or Φ_k .

The CORRAR2 Command

As in the univariate case, TIMESLAB has a command that can be used to find AR parameters from autocovariances. This command is called **CORRAR2** and the input can be either true or estimated correlations. When using estimated correlations, the user can tell **CORRAR2** to determine the best order to use by using a separate form of the command (see Section 4.2.3). The command for known order p is of the form

A=CORRAR2(R10,R20,rho120,rho1,rho2,rho12,rho21,p,sigma,ier)

where the variances of the univariate series are entered in the real scalars **R10** and **R20**, the cross-correlation of lag 0 is entered in **rho120**, while the autocorrelations for the two series and the cross-correlations of positive and negative lags are entered in the p -dimensional arrays **rho1**, **rho2**, **rho12**, and **rho21**, respectively. Finally, the input integer **p** contains the order of the AR process, while the AR coefficients and error covariance matrix are returned in the arrays **A** and **sigma** respectively. The output integer **ier** is 1 if a singular matrix is encountered in Levinson's algorithm, and 0 otherwise. If **ier** is 1, then **A** and **sigma** are not returned.

Because there are so many arguments, great care must be taken to avoid exceeding the limit of 72 characters for a command. An illustration of the use of **CORRAR2** is included in Section 4.2.3.

4.1.6. The Bivariate ARMA Process

The bivariate analog of the ARMA process is defined as the solution to the stochastic difference equation

$$\mathbf{X}(t) + \mathbf{A}(1)\mathbf{X}(t-1) + \cdots + \mathbf{A}(p)\mathbf{X}(t-p) = \boldsymbol{\epsilon}(t) + \mathbf{B}(1)\boldsymbol{\epsilon}(t-1) + \cdots + \mathbf{B}(q)\boldsymbol{\epsilon}(t-q),$$

which we can also write in terms of the AR and MA operators

$$\mathbf{G}(\mathbf{L}) = \sum_{j=0}^p \mathbf{A}(j)\mathbf{L}^j \quad \text{and} \quad \mathbf{H}(\mathbf{L}) = \sum_{k=0}^q \mathbf{B}(k)\mathbf{L}^k$$

as $\mathbf{G}(\mathbf{L})\mathbf{X}(t) = \mathbf{H}(\mathbf{L})\boldsymbol{\epsilon}(t)$. The process is stationary in terms of present and past $\boldsymbol{\epsilon}$'s if and only if the zeros of the determinant of $\mathbf{G}(z)$ are all outside the unit circle, and is invertible if and only if the zeros of the determinant of $\mathbf{H}(z)$ are all outside the unit circle. The results that we described for univariate ARMA processes have analogs in the bivariate case (see Brockwell and Davis

(1987), p. 407 for some of them), but in this book we will primarily consider the bivariate AR model.

4.1.7. The Bivariate General Linear Model

In Chapter 3 we used the univariate general linear model

$$X(t) = \sum_{k=-\infty}^{\infty} \beta_k \epsilon(t-k), \quad \epsilon \sim \text{WN},$$

as a general model for stating theorems about descriptive statistics. In this chapter we will use the bivariate analog

$$\mathbf{X}(t) = \sum_{k=-\infty}^{\infty} \mathbf{B}(k) \boldsymbol{\epsilon}(t-k),$$

where $\boldsymbol{\epsilon}$ is a bivariate white noise series.

4.2. Statistical Inferences for Bivariate Series

In this section we consider the bivariate versions of the inferences of Chapter 3. We begin by considering descriptive statistics.

4.2.1. Descriptive Statistics

Given data $\mathbf{X}(1), \dots, \mathbf{X}(n)$ from a bivariate time series \mathbf{X} , we can define the sample mean, autocovariance, and spectral density functions by

$$\bar{\mathbf{X}} = \frac{1}{n} \sum_{t=1}^n \mathbf{X}(t)$$

$$\hat{\mathbf{R}}(v) = \frac{1}{n} \sum_{t=1}^{n-|v|} (\mathbf{X}(t) - \bar{\mathbf{X}})(\mathbf{X}(t+|v|) - \bar{\mathbf{X}})^T, \quad |v| < n$$

$$\hat{\mathbf{f}}(\omega) = \sum_{v=-(n-1)}^{n-1} \hat{\mathbf{R}}(v) e^{-2\pi i v \omega}, \quad \omega \in [0, 1].$$

Note that

$$\hat{\mathbf{R}}(v) = \begin{bmatrix} \hat{R}_{11}(v) & \hat{R}_{12}(v) \\ \hat{R}_{21}(v) & \hat{R}_{22}(v) \end{bmatrix},$$

where

$$\hat{R}_{ij}(v) = \frac{1}{n} \sum_{t=1}^{n-|v|} (X_i(t) - \bar{X}_i)(X_j(t+|v|) - \bar{X}_j), \quad |v| < n,$$

while we can show (see Problem T4.9) that

$$\hat{\mathbf{f}}(\omega) = \frac{1}{n} \left(\sum_{t=1}^n (\mathbf{X}(t) - \bar{\mathbf{X}}) e^{2\pi i t \omega} \right) \left(\sum_{s=1}^n (\mathbf{X}(s) - \bar{\mathbf{X}}) e^{2\pi i s \omega} \right)^*.$$

This last equality is important for showing an important property of the bivariate sample spectral density, namely that at each frequency it is a singular matrix. We have that $\hat{\mathbf{f}}(\omega)$ is the product of a vector times its complex conjugate transpose. The rank of a product of matrices is less than or equal to the smaller of the ranks of the two matrices. In this case the matrices are vectors (which have rank at most 1) and thus $\hat{\mathbf{f}}(\omega)$ is of rank at most 1. Thus its determinant must be 0. But

$$|\hat{\mathbf{f}}(\omega)| = \hat{f}_{11}(\omega)\hat{f}_{22}(\omega) - \hat{f}_{12}(\omega)\hat{f}_{21}(\omega),$$

which means that $\hat{f}_{11}(\omega)\hat{f}_{22}(\omega) = \hat{f}_{12}(\omega)\hat{f}_{21}(\omega)$, and the sample coherence spectrum

$$\hat{W}_{12}(\omega) = \frac{|\hat{f}_{12}(\omega)|}{\sqrt{\hat{f}_{11}(\omega)\hat{f}_{22}(\omega)}} = 1, \quad \omega \in [0, 1],$$

which renders it useless in trying to estimate the true coherence.

We define the sample auto- and cross-correlation coefficients by

$$\hat{\rho}_{ij}(v) = \frac{\hat{R}_{ij}(v)}{\sqrt{\hat{R}_{ii}(0)\hat{R}_{jj}(0)}}, \quad |v| < n.$$

We next state a theorem containing the basic statistical properties of the sample mean and autocovariances. We delay consideration of the sample spectral density until the next section when we discuss smoothing it to obtain consistent spectral estimators. As in the univariate case, the results for the sample autocovariance function assume that the process mean μ either is 0 or is known and has been subtracted. The results for unknown mean and subtracting $\bar{\mathbf{X}}$ are essentially the same. Proofs of these results can be found in Hannan (1970) or Brillinger (1975), for example.

Theorem 4.2.1

 PROPERTIES OF $\bar{\mathbf{X}}$ AND $\hat{\mathbf{R}}$

Let $\mathbf{X}(1), \dots, \mathbf{X}(n)$ be a sample realization from a covariance stationary bivariate time series having autocovariance function \mathbf{R} . Then

a) If for $i, j = 1, 2$, we have $\lim_{v \rightarrow \infty} R_{ij}(v) = 0$ then

$$\lim_{n \rightarrow \infty} E(\bar{\mathbf{X}} - \mu)^T (\bar{\mathbf{X}} - \mu) = 0,$$

while if \mathbf{R} is absolutely summable, we have

$$\lim_{n \rightarrow \infty} nE(\bar{\mathbf{X}} - \boldsymbol{\mu})^T (\bar{\mathbf{X}} - \boldsymbol{\mu}) = f_{11}(0) + f_{22}(0).$$

If \mathbf{X} is a general linear process with independent errors and coefficient matrices that are absolutely summable, then

$$\bar{\mathbf{X}} \sim AN(\boldsymbol{\mu}, \frac{1}{n}\mathbf{f}(0)).$$

b) If \mathbf{X} is a general linear process with independent errors and coefficient matrices that are absolutely summable, then \hat{R}_{ij} and $\hat{\rho}_{ij}(v)$ are consistent estimators of $R_{ij}(v)$ and $\rho_{ij}(v)$. If X_1 and X_2 are univariate general linear processes with independent errors and absolutely summable coefficients, and the two univariate error series are independent, then the vector $(\hat{\rho}_{12}(j), \hat{\rho}_{12}(k))^T$ is asymptotically normal with mean zero and

$$\lim_{n \rightarrow \infty} n\text{Cov}(\hat{\rho}_{12}(l), \hat{\rho}_{12}(m)) = \sum_{v=-\infty}^{\infty} \rho_{11}(v)\rho_{22}(v+m-l).$$

If \mathbf{X} is a bivariate Gaussian process with absolutely summable autocovariance function, then the bivariate analog of Bartlett's formula is given by

$$\begin{aligned} \lim_{n \rightarrow \infty} n\text{Cov}(\hat{\rho}_{12}(l), \hat{\rho}_{12}(m)) &= \sum_{v=-\infty}^{\infty} \left[\rho_{11}(v)\rho_{22}(v+m-l) + \rho_{12}(v+m)\rho_{21}(v-l) \right. \\ &\quad - \rho_{12}(l)\{\rho_{11}(v)\rho_{21}(v+m) + \rho_{22}(v)\rho_{21}(v-m)\} \\ &\quad - \rho_{12}(v)\{\rho_{11}(v)\rho_{21}(v+l) + \rho_{22}(v)\rho_{21}(v-m)\} \\ &\quad \left. + \rho_{12}(l)\rho_{12}(m)\left\{\frac{1}{2}\rho_{11}^2(v) + \rho_{12}^2(v) + \frac{1}{2}\rho_{22}^2(v)\right\} \right]. \end{aligned}$$

If X_1 and X_2 are not cross-correlated, this reduces to

$$\text{Cov}(\hat{\rho}_{12}(l), \hat{\rho}_{12}(m)) \doteq \frac{1}{n} \sum_{v=-\infty}^{\infty} \rho_{11}(l)\rho_{22}(m),$$

which gives

$$\text{Var}(\hat{\rho}_{12}(m)) \doteq \frac{1}{n} \sum_{v=-\infty}^{\infty} \rho_{11}(v)\rho_{22}(v).$$

Implications: Part (a) allows us to find confidence regions for the vector $\boldsymbol{\mu}$. Since

$$n^{1/2}(\bar{\mathbf{X}} - \boldsymbol{\mu}) \xrightarrow{\mathcal{L}} N_2(\mathbf{0}_2, \mathbf{f}(0)),$$

we can write

$$n(\bar{\mathbf{X}} - \boldsymbol{\mu})^T \mathbf{f}^{-1}(0)(\bar{\mathbf{X}} - \boldsymbol{\mu}) \xrightarrow{L} \chi_2^2,$$

and the set of two-dimensional vectors \mathbf{h} satisfying

$$n(\bar{\mathbf{X}} - \mathbf{h})^T \mathbf{f}^{-1}(0)(\bar{\mathbf{X}} - \mathbf{h}) \leq \chi_{\alpha,2}^2$$

is a $100(1 - \alpha)\%$ confidence region for $\boldsymbol{\mu}$. The fact that $\mathbf{f}(0)$ is unknown can be overcome by using a consistent estimate of it. For large samples, this has no effect on the confidence level.

Part (b) of the theorem contains the crucial fact that the variance of the sample cross-correlation coefficient is a function of the autocorrelation functions of the two series. For example, if X_1 and X_2 are independent AR(1) processes with coefficients α and β respectively, then we have

$$\rho_{11}(v) = (-\alpha)^v \quad \text{and} \quad \rho_{22}(v) = (-\beta)^v,$$

which means that

$$\text{Var}(\hat{\rho}_{12}(v)) \doteq \frac{1}{n} \frac{1}{1 - \alpha\beta}.$$

This gives an approximate 95% confidence interval for $\rho_{12}(v)$ as

$$\hat{\rho}_{12}(v) \pm \frac{2}{\sqrt{n(1 - \alpha\beta)}}.$$

For $n = 100$, $\alpha = -.9$, and $\beta = -.8$, this interval is $\hat{\rho}_{12}(v) \pm 0.38$, which means that a sample cross-correlation as large as 0.38 would not be surprising. See Problem C4.3 for another example.

Testing for Independence

We have seen that two univariate series that are jointly covariance stationary are uncorrelated if their cross-correlation function is zero for all lags. If the bivariate series is Gaussian, then this also means that the two series are independent. Thus a natural method for testing for independence of two univariate series is to test the hypothesis that their cross-correlations are zero. Unfortunately, we have seen that the sample cross-correlations can appear to be quite large when the true cross-correlations are zero if there is high autocorrelation in the univariate series. Traditionally, the solution to this problem has been to use the cross-correlations for a filtered version of the original series rather than the series themselves. This process of filtering the data prior to analyzing them is called “prewhitening” the data. The justification for prewhitening when testing for independence is given in the next theorem.

Theorem 4.2.2

TESTING FOR INDEPENDENCE

The jointly covariance stationary univariate time series X_1 and X_2 are uncorrelated (and thus independent if they are Gaussian) if and only if the series $Y_1(t) = g_1(L)X_1(t)$ and $Y_2(t) = g_2(L)X_2(t)$ are uncorrelated, where g_1 and g_2 are any two invertible filters.

Proof: We can write $\mathbf{Y}(t) = \mathbf{G}(L)\mathbf{X}(t)$, where

$$\mathbf{G}(L) = \begin{bmatrix} g_1(L) & 0 \\ 0 & g_2(L) \end{bmatrix},$$

and thus by the bivariate Filter Theorem we have

$$\mathbf{f}_Y(\omega) = \mathbf{G}(e^{2\pi i\omega})\mathbf{f}_X(\omega)\mathbf{G}^*(e^{2\pi i\omega}).$$

Now X_1 and X_2 are uncorrelated if and only if $\mathbf{f}_X(\omega)$ is diagonal for all ω since \mathbf{f}_X and \mathbf{R}_X are Fourier transforms of each other. But since \mathbf{G} is diagonal, we have that \mathbf{f}_X is diagonal if and only if \mathbf{f}_Y is.

Thus a simple test for independence of two jointly Gaussian covariance stationary time series is to (1) fit autoregressive processes to the univariate series, (2) calculate the cross-correlations of the errors, and (3) test whether these cross-correlations are significantly different from zero. Under the null hypothesis, the cross-correlations are asymptotically independent and identically distributed with mean zero and asymptotic variance $1/n$. Thus if we test each of $2M + 1$ cross-correlations (usually for lags $-M$ to M), we should use a significance level of $\alpha^{1/2M+1}$ for each lag so that the overall significance level will be α .

In Example 4.3 we give a macro for carrying out this procedure. In Figure 4.4 we display the result of applying this macro to the gas furnace data set introduced in Chapter 1. The data set itself is also given in Figure 4.4. Note that we multiplied X_1 by three and then added 30 so that the features of the data can be seen. The values of `m`, `alpha`, and `maxp` that were used were 20, .05, and 10 respectively. The autoregressive models for the univariate series have orders 6 and 4 and coefficients

$$\alpha_1 = (-1.93, 1.20, -0.19, 0.13, -0.27, 0.11)^T$$

$$\alpha_2 = (-1.85, 0.84, 0.31, -0.26)^T.$$

All of the cross-correlations fall within the bands except those for lags 3–6. Thus we can conclude that the two series are not independent. If we look at

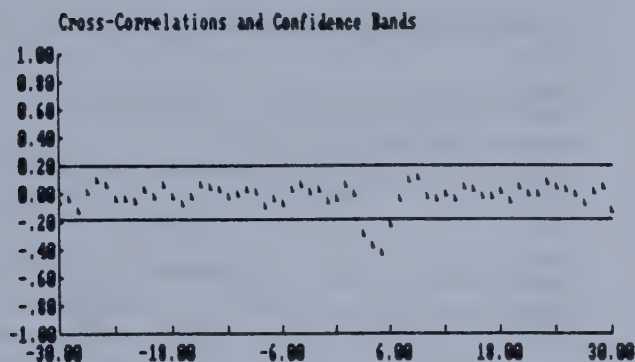


Figure 4.4. The Gas Furnace Data and the Cross-Correlations and 95% Confidence Bands for the Prewhitened Data.

the data themselves, we can see that the input series is leading the output series by approximately 3-6 time periods.

4.2.2. Nonparametric Spectral Density Estimation

Given data from a bivariate time series, we can apply the methods of Section 3.2 to the sample autospectral densities \hat{f}_{11} and \hat{f}_{22} to obtain consistent estimates of f_{11} and f_{22} . In this section we consider applying windows to the sample cross-spectral density \hat{f}_{12} . Traditionally, the same smoothing is applied to all three functions \hat{f}_{11} , \hat{f}_{22} , and \hat{f}_{12} so that the sampling properties of the resulting smoothed estimators can be easily stated.

The CROSSP Command

The command

```
CROSSP(rho12,rho21,R10,R20,rho120,M,Q,kernel,f12r,f12i)
```

will calculate the real and imaginary parts of the smoothed cross-spectra at the $q = [Q/2] + 1$ natural frequencies using the truncation point entered in **M** and the kernel determined by the input integer **kernel**, which can be any one of the first five kernels as described in the **WINDOW** command.

In Example 4.4 we give a macro that will calculate and display the coherence and phase spectra (together with the confidence intervals that we derive below) for a bivariate data set. The results of applying this to the prewhitened gas furnace data for the Parzen kernel having truncation point 36 are displayed in Figure 4.5. We will discuss the implications of these graphs later, but for now, note that the coherence is highest for low frequencies. Also note that the

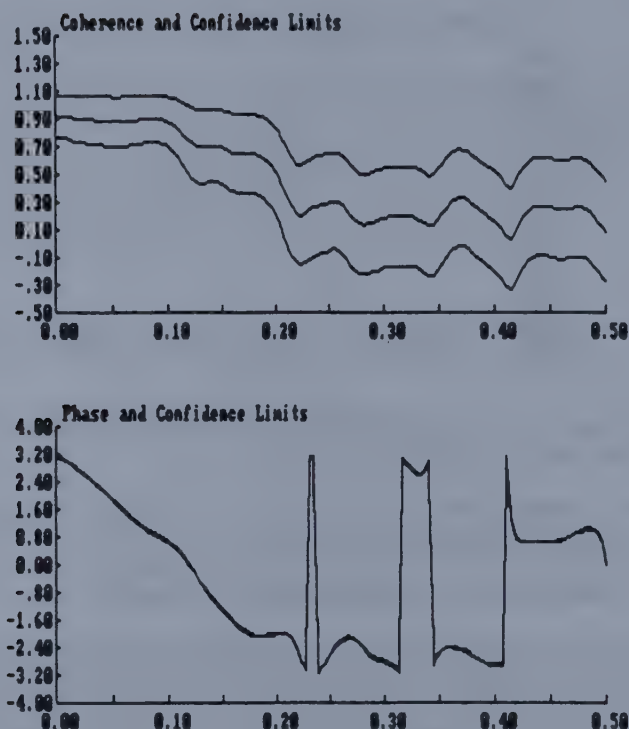


Figure 4.5. Coherence and Phase Spectra and Confidence Intervals for the Prewhitened Gas Furnace Data Using the Parzen Window with Truncation Point 36.

phase spectrum is smooth in pieces, with a sudden jump from the top of the graph to the bottom. This is because the arctangent is taken modulo 2π , and thus when the curve reaches the top of the graph, it will “wrap around” to the bottom.

Sampling Properties of Sample Cross-Spectral Density

In Section 3.1 we saw that the sample autospectral densities are asymptotically uncorrelated at different frequencies and are asymptotically proportional to χ^2 random variables. In the extension to the bivariate case, we have two further questions: (1) How are \hat{f}_{11} and \hat{f}_{22} related, and (2) what is the large sample behavior of the complex valued random variable $\hat{f}_{12}(\omega)$? Notice that in the matrix $\hat{\mathbf{f}}(\omega)$ there are a total of four distinct, real, scalar random variables,

namely $\hat{f}_{11}(\omega)$, $\hat{f}_{22}(\omega)$, $\hat{c}_{12}(\omega)$, and $\hat{q}_{12}(\omega)$, since

$$\hat{f}_{12}(\omega) = \hat{c}_{12}(\omega) - i\hat{q}_{12}(\omega) \quad \text{and} \quad \hat{f}_{21}(\omega) = \hat{c}_{12}(\omega) + i\hat{q}_{12}(\omega).$$

We have the following results about these four random variables.

Theorem 4.2.3 PROPERTIES OF BIVARIATE PERIODOGRAM

Let $\hat{\mathbf{f}}$ be the bivariate sample spectral density function for a realization of length n from a general linear process having independent errors and coefficients satisfying

$$\sum_{k=-\infty}^{\infty} |B_{ij}(k)| |k|^{1/2} < \infty, \quad i, j = 1, 2.$$

If $0 < \omega_j < \omega_k < 0.5$ are either fixed frequencies or two natural frequencies, then the elements of $\hat{\mathbf{f}}(\omega_j)$ are asymptotically uncorrelated with the elements of $\hat{\mathbf{f}}(\omega_k)$, and if we let

$$\hat{\mathbf{Z}}(\omega) = (\hat{f}_{11}(\omega), \hat{f}_{22}(\omega), \hat{c}_{12}(\omega), \hat{q}_{12}(\omega))^T$$

for $0 < \omega < 0.5$, then (omitting the argument ω from each element)

$$\begin{aligned} & \lim_{n \rightarrow \infty} \text{Var}(\hat{\mathbf{Z}}(\omega)) \\ &= \begin{bmatrix} f_{11}^2 & |f_{12}|^2 & f_{11}c_{12} & f_{11}q_{12} \\ & f_{22}^2 & f_{22}c_{12} & f_{22}q_{12} \\ & & \frac{1}{2}[f_{11}f_{22} + c_{12}^2 - q_{12}^2] & c_{12}q_{12} \\ \text{Symmetric} & & & \frac{1}{2}[f_{11}f_{22} - c_{12}^2 + q_{12}^2] \end{bmatrix} \\ &\equiv \mathbf{V}. \end{aligned}$$

We will not consider the asymptotic distribution of the cross-spectral density as we will use smoothed estimates to make inferences about the coherence, gain, and phase spectra. For this we have the following theorem.

Theorem 4.2.4

BIVARIATE WINDOW ESTIMATORS

Let

$$\tilde{\mathbf{f}}(\omega) = \sum_{v=-\infty}^{\infty} \lambda\left(\frac{v}{M}\right) \hat{\mathbf{R}}(v) e^{-2\pi i v \omega}$$

be a smoothed periodogram estimator of $\mathbf{f}(\omega)$ based on a realization of length n from a general linear process having independent errors and coefficients that are absolutely summable. If $M \rightarrow \infty$ and $M/n \rightarrow \infty$ as $n \rightarrow \infty$, and $0 < \omega_j < \omega_k < 0.5$, then the elements of $\tilde{\mathbf{f}}(\omega_j)$ are asymptotically uncorrelated with the elements of $\tilde{\mathbf{f}}(\omega_k)$. If we define

$$\tilde{\mathbf{Z}}(\omega) = (\tilde{f}_{11}(\omega), \tilde{f}_{22}(\omega), \tilde{c}_{12}(\omega), \tilde{q}_{12}(\omega))^T$$

$$\mathbf{Z}(\omega) = (f_{11}(\omega), f_{22}(\omega), c_{12}(\omega), q_{12}(\omega))^T,$$

then

$$\sqrt{\frac{n}{M}}(\tilde{\mathbf{Z}}(\omega) - \mathbf{Z}(\omega)) \xrightarrow{\mathcal{L}} N_4(\mathbf{0}_4, \gamma \mathbf{V}),$$

where the asymptotic covariance matrix \mathbf{V} is given in Theorem 4.2.3 and

$$\gamma = \int_{-\infty}^{\infty} \lambda^2(u) du.$$

This theorem allows us to make inferences about the autospectra and the co- and quadrature spectra. It also allows us to find by the continuous function theorem (see Theorem A.4.5) the asymptotic distribution of the quantities (such as coherence and phase) that are functions of the elements of $\tilde{\mathbf{Z}}$. These are given in the next theorem.

Theorem 4.2.5

PROPERTIES OF SPECTRAL QUANTITIES

Under the conditions of Theorem 4.2.4, we have:

a) If the squared coherence is strictly positive at frequency ω , then the smoothed estimate of the amplitude spectrum satisfies

$$\sqrt{n/M} \left(\tilde{A}_{12}(\omega) - A_{12}(\omega) \right) \xrightarrow{\mathcal{L}} N(0, \sigma_A^2(\omega)),$$

where

$$\sigma_A^2(\omega) = \frac{\gamma A_{12}^2(\omega)}{2} \left(1 + \frac{1}{W_{12}^2(\omega)} \right).$$

b) If the squared coherence is strictly positive at frequency ω , then the smoothed estimate of the phase spectrum satisfies

$$\sqrt{n/M} \left(\tilde{\phi}_{12}(\omega) - \phi_{12}(\omega) \right) \xrightarrow{\mathcal{L}} N(0, \sigma_{\phi}^2(\omega)),$$

where

$$\sigma_{\phi}^2(\omega) = \frac{\gamma A_{12}^2(\omega)}{2} \left(\frac{1}{W_{12}^2(\omega)} - 1 \right).$$

If $W_{12}^2(\omega) = 0$, then $\tilde{\phi}_{12}(\omega)$ is asymptotically uniformly distributed over the interval $(-\pi, \pi)$.

c) If the squared coherence is strictly positive at frequency ω , then the smoothed estimate of the coherence spectrum satisfies

$$\sqrt{n/M} \left(\tilde{W}_{12}(\omega) - W_{12}(\omega) \right) \xrightarrow{\mathcal{L}} N(0, \sigma_W^2(\omega)),$$

where

$$\sigma_W^2(\omega) = \frac{\gamma}{2} (1 - W_{12}^2(\omega)).$$

The small sample bias of the coherence estimate can be appreciable if X_1 and X_2 have a large cross-correlation for nonzero lags.

d) If $W_{12}(\omega) = 0$, then

$$\frac{c |\tilde{W}_{12}(\omega)|^2}{(1 - |\tilde{W}_{12}(\omega)|^2)} \sim F_{2, 2c},$$

where $c = (n/M\gamma) - 1$.

Implications: These results provide large sample confidence intervals for the amplitude, phase, and coherence spectra given by

$$\begin{aligned} \tilde{A}_{12}(\omega) &\pm Z_{\alpha/2} \sqrt{\frac{M}{n} \tilde{\sigma}_A^2(\omega)} \\ \tilde{\phi}_{12}(\omega) &\pm Z_{\alpha/2} \sqrt{\frac{M}{n} \tilde{\sigma}_{\phi}^2(\omega)} \\ \tilde{W}_{12}(\omega) &\pm Z_{\alpha/2} \sqrt{\frac{M}{n} \tilde{\sigma}_W^2(\omega)}, \end{aligned}$$

where the $\tilde{\sigma}^2$'s are evaluated by substituting estimates of unknown quantities into the formulas for the σ^2 's. Part (d) provides an easily applied test for zero coherence at individual frequencies. To illustrate this, we have Figure

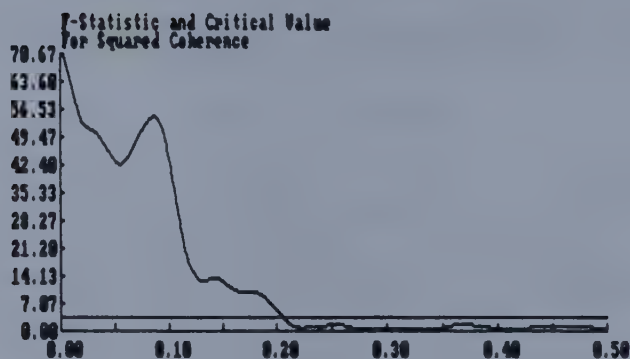


Figure 4.6. Transformed Squared Coherence and Critical Value for the F Test for the Prewhitened Gas Furnace Data.

4.6, which consists of a graph of the transformed squared coherence for the prewhitened gas furnace data. Superimposed on the graph is the critical value of the F distribution. See Example 4.5 for the macro that generated this figure.

4.2.3. Autoregressive Spectral Estimation and Forecasting

As in the univariate case, autoregressive modeling is easily applied and as long as the data being analyzed come from a process which can be expressed as an infinite order AR process, one can use the chosen model for spectral estimation or forecasting.

The CORRAR2 Command

If the order of the process to be fit is known, then the known-order form of the CORRAR2 command described in Section 4.1.5 can be used to find parameter estimates. If the order is unknown, then the command

```
A=CORRAR2(RX0,RY0,rhoxy0,rhox,rhoy,rhoxy,rhoxy,M,iopt,
n,p,SIGMA,cat,ier)
```

can be used to estimate the order p and the parameters of the process. This form is the same as the known-order form except that M is the maximum order to fit, n is the sample size, cat is an output array of length M containing the

values of the bivariate CAT criterion (see Parzen (1977))

$$\text{CAT}(k) = \text{tr} \left[\left(\frac{2}{n} \sum_{j=1}^k \tilde{\Sigma}_j^{-1} \right) - \tilde{\Sigma}_k^{-1} \right],$$

where $\tilde{\Sigma}_j$ is the “unbiased residual variance” of order j

$$\tilde{\Sigma}_j = \frac{n}{n - 2j} \hat{\Sigma}_j$$

and $\hat{\Sigma}_j$ is the order- j “biased residual variance” estimator

$$\hat{\Sigma}_j = \sum_{l=0}^j \hat{A}_j(l) \hat{R}(l).$$

The input integer `iopt` is 1 or 2 depending on whether order `M` is to be used or the CAT criterion determined order less than or equal to `M` is to be used.

Once the order and parameters of the AR process have been determined, then the `ARSF2` command can be used to find the corresponding autoregressive spectral estimator. In addition, forecasts of future values of the process can be found by

$$\hat{X}(n + h) = - \sum_{j=1}^p \hat{A}(j) \hat{X}(n + h - j),$$

where $\hat{X}(n + h - j) = X(n + h - j)$ if $j \geq h$. In Example 4.6 we give a macro that (1) calculates and displays estimates of the auto- and cross-correlations from a data set, (2) fits a bivariate AR model to the data, and (3) uses this model to find autospectral estimates as well as estimates of the coherence and phase spectra. We applied this macro to both the original gas furnace data and the prewhitened gas furnace data and found the following correlations and AR models:

Original Gas Furnace Data				
v	rho1(v)	rho2(v)	rho12(v)	rho21(-v)
0.	1.000	1.000	-.484	-.484
1.	.952	.971	-.598	-.393
2.	.834	.896	-.725	-.329
3.	.682	.793	-.843	-.286
4.	.531	.680	-.925	-.260
5.	.408	.574	-.950	-.243
6.	.318	.485	-.915	-.227
7.	.260	.416	-.829	-.206
8.	.228	.366	-.717	-.179
9.	.213	.330	-.600	-.149

```

10.      .208      .307      -.495      -.118
Autoregressive Coefficient Matrices
-1.925889      .001240
-.050508      -1.299794
1.201669      -.004222
.020493      .327703
-.116929      .008672
.711791      .257010
-.104232      -.003270
-.195398      -.133420
Autoregressive Error Variance Matrix
.035155      -.007313
-.007314      .097240

```

Preshitened Data

```

-----
v      rho1(v)      rho2(v)      rho12(v)      rho21(-v)
-----
0.      1.000      1.000      -.041      -.041
1.      .010      .269      .052      -.060
2.      .013      .083      .000      .019
3.      -.010      -.001      -.288      .007
4.      -.042      -.090      -.368      .064
5.      -.004      -.061      -.434      .031
6.      .077      .094      -.227      -.085
7.      .021      .057      -.037      -.044
8.      -.074      .011      .095      -.087
9.      -.090      -.068      .113      .013
10.     .035      .007      -.026      .025

```

```

Autoregressive Coefficient Matrices
-.008204      .030892
-.072571      .303707
-.024308      -.036640
.019692      .303947
.005909      .000587
.555159      .169584
.051469      -.027097
.853693      .153809
.013523      -.006732
1.135349      .121737
-.077010      .048511
.921561      -.050637
-.051536      -.010818
.647568      -.038759
.049215      .041825
.311347      -.042690
Autoregressive Error Variance Matrix
.033566      -.001345
-.001345      .054038

```

The prewhitening has indeed removed the autocorrelation in the two data sets, while the cross-correlation remains high for lags 3-6. An important thing to notice is that the prewhitening has greatly decreased the cross-correlation for negative lags, and the cross-correlations for the prewhitened data clearly

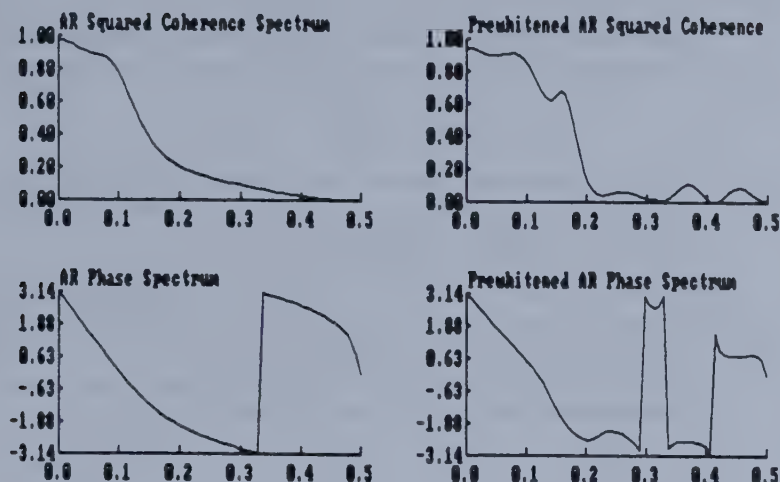


Figure 4.7. Phase and Squared Coherence Estimates for the Original and Prewhitened Gas Furnace Data.

reveal that the dependence of the output to the input is as a delay process. In Figure 4.7, we give the squared coherence and phase estimates for the two data sets. The linear shape of the phase confirms the delay nature of the dependence (see Section 4.1.4). Note that one should also estimate the coherence after the two series have been aligned (see Problem C4.4), because as stated in part (c) of Theorem 4.2.5, the presence of large cross-correlations at nonzero lags can cause a bias in the coherence estimator. In any event, the coherence is clearly significantly different from zero, and in the next section we discuss modeling the dependence of the output series on the input series.

4.2.4. Estimating the Transfer Function of a Filter

In Section 4.1.4 we considered approximating the optimal filter for one univariate series based on another in the case where the true auto- and cross-spectra were known for the series. In this section we consider substituting estimates of the spectra into the procedures for the known spectra case. For another approach to the problem of estimating filter coefficients, see Chapter 11 of Box and Jenkins (1970).

Given data $X(1), \dots, X(n)$ from a covariance stationary bivariate time series \mathbf{X} , we consider the following procedure for expressing $X_2(t)$ as a filtered version of X_1 :

1. Prewhiten X_1 and X_2 using univariate, CAT criterion determined order

AR filters. Call the two orders \hat{p}_1 and \hat{p}_2 and the corresponding polynomials \hat{g}_1 and \hat{g}_2 . Thus the result of this stage is the two error series

$$e_1(t) = \hat{g}_1(L)X_1(t) \quad \text{and} \quad e_2(t) = \hat{g}_2(L)X_2(t).$$

If the two orders are different, then we align the two error series so that the result is a bivariate series e of length $n - \max(\hat{p}_1, \hat{p}_2)$. The macro PREWHITE (see Example 4.3) performs this procedure.

2. In this step we use either nonparametric or autoregressive spectral estimation to find estimates of the auto- and cross-spectra of the e series. We then substitute these estimators into the procedure (see the FIND-FILT macro in Section 4.1.4) for approximating the integrals that find the coefficients of the filter for e_2 as a function of e_1 .
3. The effect of the first two steps of the procedure is to write

$$\hat{g}_2(L)X_2(t) = \hat{h}(L)\hat{g}_1(L)X_1(t),$$

where $\hat{h}(L)$ is the polynomial of the filter found in step 2. Thus we have

$$X_2(t) = \hat{g}_2^{-1}(L)\hat{h}(L)\hat{g}_1(L)X_1(t).$$

The final step of the process is to multiply the three polynomials together to obtain the final filter expressing X_2 as a function of X_1 .

The following macro will carry out this process. To use it, the user must enter the input and output series and their length in the variables x , y , and n , the maximum AR order to be allowed in the prewhitening procedure in maxp , the number of frequencies and truncation point for the nonparametric estimator in Q and nt , and finally the number of filter coefficients to find in m .

```

1 ;;
2 ;;   FILTEST.MAC: This macro estimates the filter coefficients
3 ;;       relating an output series y to an input series x.
4 ;;       It is assumed that only coefficients with nonnegative
5 ;;       lags are in the model.
6 ;;
7 ;;       The series are first prewhitened (via the PREWHITE
8 ;;       macro). Then the auto- and cross-spectra are found
9 ;;       for the prewhitened series. Then the FINDFILT macro
10 ;;       is used to estimate the filter coefficients.
11 ;;       Then the filter for the original data is found.
12 ;;
13 ;;   INPUT: x,y,n (original series and their length)
14 ;;       maxp (maximum AR lag for prewhitening)
15 ;;       Q,nt (number of frequencies and truncation point in spectra)
```

```

16 ;;          m (number of filter coefficients to find)
17 ;;
18 PAUSE
19 ;start
20 ;
21 PROMPTOFF
22 MACRO(white,start)
23 ;
24 CORR2(xx,yy,nn,nt,1,rx0,ry0,rhoxy0,rhox,rhoxy,rhoxy,rhoxy)
25 fxx=WINDOW(rhox,rx0,nt,Q,4)
26 fyy=WINDOW(rhoxy,ry0,nt,Q,4)
27 CROSSP(rhoxy,rhoxy,rx0,ry0,rhoxy0,nt,Q,4,fxyr,fxyi)
28 ;
29 MACRO(findfilt,start)
30 ;
31 beta=beta/beta0
32 gama=MULTPOLY(beta,alphax,m,px)
33 mppx=m+px
34 betay=INVLPOLY(alphay,py,40)
35 gama=MULTPOLY(betay,gama,40,mppx)
36 gama=gama*beta0
37 LABEL(gama)='Final Filter Coefficients'
38 LIST(gama,m)
39 list(beta0)
40 PROMPTON

```

To illustrate the use of this macro, we used the gas furnace data with $\max p=10$, $Q=128$, $nt=30$, and $m=10$, and found the following filter coefficients:

```

Coefficients From FILTEST for Gas Furnace Data (beta0=-0.069691)
1 | .084685  -.036375  -.495704  -.591778  -.712478
6 | -.411839  -.285202  -.099515  -.042972  -.061627

```

These agree rather well with the model found by Box and Jenkins (1970), p. 399 using parametric methods:

$$(1 - 0.57L)X_2(t) = -(0.53 + 0.37L + 0.51L^2)X_1(t - 3),$$

since if we invert $1 - 0.57L$ and multiply it times the polynomial on the right hand side of the model, we get

```

Coefficients of Box-Jenkins Model For Gas Furnace Data (beta0=0)
1 | .000000  .000000  -.530000  -.672100  -.893097
6 | -.509065  -.290167  -.165395  -.094275  -.053737

```

See Problem C4.5 for another illustration of the use of the FILTEST macro.

4.3. Other Topics in Bivariate and Multiple Time Series

In the first three chapters of this book we presented a wide variety of models and estimation procedures for univariate time series. In this chapter we have emphasized nonparametric and autoregressive spectral estimation and autoregressive prediction. These methods are well established and are widely used in science. It is hoped that as personal computers become more powerful (particularly in the area of having access to more random access memory), we can revise TIMESLAB to incorporate more of the analogs from the univariate case, as well as extend the bivariate commands to become multivariate commands. Most of the ideas of bivariate time series analysis extend directly to the multiple case, where instead of observing only two variables at each time one observes $d > 2$ variables. Thus the autocovariance and spectral density functions become $(d \times d)$ matrices, as do AR coefficients and filter coefficients in general. The books by Priestley (1981) and Brillinger (1975) provide a good description of the general area of multiple time series.

4.4. Examples and Problems

Example 4.1

FORMING TABLE 4.1 AND FIGURES 4.1 AND 4.2

The data and correlations that are discussed in Section 4.1.1 come from a bivariate autoregressive model (see Section 4.1.5). The `ARCORR2` command can be used to find the correlations for such a process while the `ARDT2` command can be used to simulate a realization. The macro `FIG41.MAC` was used to form Table 4.1 and Figure 4.1:

```

1 ;;
2 ;;   FIG41.MAC: macro to form Figure 4.1
3 ;;
4 ;;   INPUT: none
5 ;;
6 PAUSE
7 ;start
8 a=<-.6,-.4,.5,-.5>
9 sigma=<1,0,0,1>
10 M=12
11 ARCORR2(a,1,sigma,M,r10,r20,rho120,rho1,rho2,rho12,rho21,ier)
12 MACRO(corr2plt,start)   ;produce table and plots
13 ;
14 ;   Generate and Plot Realizations:
15 ;
16 ;data
17 x1=ARDT2(a,1,sigma,0,60,ier)
18 x=EXTRACT(x1,1,120,2,n) ;every other one starting with 1st is X1
19 y=EXTRACT(x1,2,120,2,n) ;every other one starting with 2nd is X2

```

```

20 n=60
21 y=y-4
22 x=x+4
23 yy=<y,x>
24 ii=LINE(60,0,1)
25 xx=<ii,ii>
26 type=<12,12>
27 LABEL(yy)='Realization From Bivariate Process'
28 LABEL(xx)='X1 is on the Top'
29 PLOTK(xx,yy,60,2,type,0,60,-10,10)

```

Note that FIG41 uses the macro CORR2PLT.MAC to actually produce a table of correlations (which was saved in a file using the RECORD command, edited slightly, and then inserted into the file that produced this book) and the graphs of correlations and data sets:

```

1 ;;
2 ;;  CORR2PLT.MAC: macro to display auto and cross-correlations
3 ;;
4 ;;  INPUT: rho120 (cross-correlation of lag 0)
5 ;;          M (number of lags at which correlations have been found)
6 ;;          rho1, rho2 (arrays containing autocorrelations)
7 ;;          rho12, rho21 (arrays containing cross-correlations)
8 ;;
9 ;;
10 PAUSE
11 ;start
12 ;
13 ;  Construct the Table of Correlations:
14 ;
15 mp1=M+1
16 ind=LINE(mp1,-1,1)
17 rr=<ind,1,rho1,1,rho2,rho120,rho12,rho120,rho21>
18 rr=TRANS(rr,mp1,5)
19 LABEL(rr)='Table of Auto and Cross Correlations'
20 mp15=mp1*5
21 form='f3.0,4f9.3'
22 LIST(rr,mp15,5,form)
23 PAUSE
24 ;
25 ;  Autocorrelation Plots:
26 ;
27 PLOT(rho1,M,0,M,-1,1)
28 PLOT(rho2,M,0,M,-1,1)
29 ;
30 ;  Construct the Cross-Correlation Plot:
31 ;
32 ind=LINE(M,0,1)
33 ii=-ind
34 ii=REVERSE(ii,M)
35 ind=<ii,0,ind>

```



```

36 rr=rho21
37 rr=REVERSE(rr,M)
38 rr=<rr,rho120,rho12>
39 rr=<rr,-1,1,0,0,1,1,1,-1>
40 MM=-M
41 m2p1=2*M
42 m2p1=m2p1+1
43 ind=<ind,0,0,MM,M,MM,M,M,M>
44 nn=<m2p1,2,2,2,2>
45 type=<13,2,2,2,2>
46 LABEL(rr)='Cross-Correlations'
47 LABEL(ind)='      '
48 PLOTK(ind,rr,nn,5,type,MM,M,-1,1)

```

Notice how the PLOTK command is used in FIG41 and CORR2PLT. Of particular interest is how the vertical and horizontal lines are superimposed on the cross-correlation plot, and how the individual points on the data plot can be seen by using `type=<12,12>` (see the PLOTK command). Note also that the ARDT2 command generates a bivariate realization of length n in an array of length $2n$, in which the elements of the first univariate series are the odd numbered elements and the second univariate series are the even numbered ones. The EXTRACT command is used to form individual arrays for the two univariate series. To obtain Figure 4.2, we used the ARSP2 command to find auto- and cross-spectra and then used the macro CROSSP.MAC given in Section 4.1.2 to calculate and plot the coherence and phase spectra.

Example 4.2

INTERPRETING COHERENCE

The macro COHER.MAC was used to perform the experiment in Section 4.1.2 that illustrated the interpretation of the coherence function. Notice how the frequency for a scatterplot is inserted into its plotting label.

```

1 ;;
2 ;;   COHER.MAC : This macro generates nsamps samples of length n
3 ;;               for a bivariate AR process and for each one finds
4 ;;               the univariate periodograms at the n frequencies
5 ;;               between 0 and 1. A scatterplot of the amplitudes
6 ;;               for frequency (nf-1)/n is then produced.
7 ;;
8 ;;   INPUT: n,nsamps,nf,seed
9 ;;
10 ;;
11 PAUSE
12 ;start
13 PROMPTOFF
14 a=<-.6,-.4,.5,-.5>
15 sigma=<1,0,0,1>
16 c1=LINE(nsamps)      ;reserve space for amplitudes

```

```

17 c2=LINE(nsamps)
18 x=WN(seed,10)           ;warm up random number generator
19 ns=1                     ;initialize sample counter
20 ;
21 ;s1
22 ;
23 LIST(ns)
24 x=ARDT2(a,1,sigma,0,n,ier) ;generate bivariate AR sample
25 n2=2*n
26 x1=EXTRACT(x,1,n2,2,nx)   ;pull off the univariate series
27 x2=EXTRACT(x,2,n2,2,nx)
28 rho=CORR(x1,n,0,n,1,r01,f1) ;find periodograms
29 rho=CORR(x2,n,0,n,1,r02,f2)
30 c1[ns]=f1[nf]             ;pull off amplitude for desired frequency
31 c2[ns]=f2[nf]
32 IF(ns.eq.nsamps,end)
33 ns=ns+1
34 GOTO(s1)
35 ;
36 ;end
37 ;
38 mnsamps=-nsamps
39 LABEL(c2)='Scatterplot of Amplitudes'
40 LABEL(c1)=' '
41 PLOT(c1,c2,mnsamps)       ;do scatterplot

```

Example 4.3	TESTING FOR INDEPENDENCE
--------------------	--------------------------

The macro INDTEST.MAC can be used to test for independence of two univariate time series as described in and below Theorem 4.2.2.

```

1 ;;
2 ;; INDTEST.MAC : Macro to test two univariate series for
3 ;; independence. AR processes are fit to each
4 ;; series and the cross-correlations of lags
5 ;; -m to m graphed with 100(1-alpha)%
6 ;; confidence bands.
7 ;;
8 ;; INPUT: x, y, n (the two series and their length), m,alpha
9 ;; maxp (maximum AR order allowed)
10 ;;
11 PAUSE
12 ;start
13 MACRO(prewrite,start)
14 CORR2(xx,yy,nn,m,1,rx0,ry0,rhoxy0,rhox,rhoxy,rhoxy,rhoxy)
15 mm=-m
16 rhoxy=REVERSE(rhoxy,m)
17 rho=<rhoxy,rhoxy0,rhoxy> ;rho has the correlations for lags -m to m
18 ;
19 ; Find Z-value to use:
20 ;

```



```

31      n1=nx-ny
32      n1=n1+1
33      xx=EXTRACT(xx,n1,nx)
34  ENDIF
35  IF(nx.lt.ny)      ;handle case where y residuals longer
36      nn=nx        ;(delete first part of y residuals)
37      n1=ny-nx
38      n1=n1+1
39      yy=EXTRACT(yy,n1,ny)
40  ENDIF
41  ;
42  ;end
43  ;

```

Example 4.4 NONPARAMETRIC SPECTRAL ESTIMATION

The macro CROSSPE.MAC can be used to find nonparametric estimates of the phase spectrum and squared coherency spectrum and confidence intervals for each given a data set.

```

1  ;;
2  ;;  CROSSPE.MAC: This macro will compute the smoothed coherence and
3  ;;      phase spectrum for two series and put confidence
4  ;;      intervals on them. These are not confidence bands.
5  ;;
6  ;;  INPUT: x,y,n (two series and their length)
7  ;;      m, kernel (scale parameter and kernel option)
8  ;;      Q (number of frequencies between 0 and 1)
9  ;;
10 PAUSE
11 ;start
12 gamm=<2,.66667,.795,.539,.586> ;see Table 3.3
13 CORR2(x,y,n,m,1,RXO,RYO,rhoxy0,rhox,rhoy,rhoxy,rhoxyx)
14 fxx=WINDOW(rhox,RXO,m,Q,kernel)
15 fyy=WINDOW(rhoy,RYO,m,Q,kernel)
16 CROSSP(rhoxy,rhoxyx,RXO,RYO,rhoxy0,m,Q,kernel,fxyr,fxyi)
17 q=(Q/2)+1
18 freq=LINE(q,0,.5,1) ;get frequencies for plots
19 POLAR(fxyr,fxyi,q,amp,phase) ;now have phase spectra
20 coher=amp/{fxx*fyy}^ .5 ;coherence
21 ;
22 ; Do coherence:
23 ;
24 se={({m*gamm[kernel]}/{2.*n})*{1-coher^2}}^ .5
25 cohlow=coher-2*se
26 cohup=coher+2*se
27 ff=<freq,freq,freq>
28 cc=<coher,cohlow,cohup>
29 label(cc)='Coherence and Confidence Limits'
30 LABEL(ff)= ' '

```

```

31 lengs=<q,q,q>
32 type=<2,2,2>
33 PLOTK(ff,cc,lengs,3,type,0,.5,-.5,1.5)
34 ;
35 ;   Do phase:
36 ;
37 fac=amp^2*(1-coher^2)/{coher^2}
38 se={m*gamma[kernel]/{2.*n}}*fac^.5
39 phlow=phase-2*se
40 phup=phase+2*se
41 cc=<phase,phlow,phup>
42 LABEL(cc)='Phase and Confidence Limits'
43 PLOTK(ff,cc,lengs,3,type,0,.5,-4,4)

```

Example 4.5**TESTING FOR ZERO COHERENCE**

In Example 4.3 we gave a macro that can be used to test for independence of two univariate series based on the cross-correlation function. The macro COHTEST.MAC provides a test for independence based on the squared coherence function. This test is based on part (d) of Theorem 4.2.5.

```

1 ;;
2 ;;   COHTEST.MAC: This macro will test the hypothesis of 0
3 ;;               coherence at each of the [Q/2]+1
4 ;;               frequencies between 0 and .5.
5 ;;
6 ;;   INPUT: x,y,n (two series and their length)
7 ;;           m, kernel (scale parameter and kernel option)
8 ;;           Q (number of frequencies between 0 and 1)
9 ;;
10 PAUSE
11 ;start
12 gamm=<2,.66667,.795,.539,.586> ;see Table 3.3
13 CORR2(x,y,n,m,1,RX0,RYO,rhoxy0,rhox,rhoxy,rhoxy,rhoxy)
14 fxx=WINDOW(rhox,RX0,m,Q,kernel) ;window auto- and cross-spectra
15 fyy=WINDOW(rhoxy,RYO,m,Q,kernel)
16 CROSSP(rhoxy,rhoxy,RX0,RYO,rhoxy0,m,Q,kernel,fxyr,fxyi)
17 ;
18 ;plot F-test:
19 ;
20 w2={fxyr^2+fxyi^2}/{fxx*fyy} ;squared coherence
21 c={n-m*gamma[kernel]}/{m*gamma[kernel]} ;
22 fstat={c*w2}/{1-w2}
23 nc:=2*c ;round off degrees of freedom
24 ft=DIST(f,3,1,.95,2,nc) ;find critical value
25 q={Q/2}+1 ;number of frequencies
26 freq=LINE(q,0,.5,1) ;get frequencies for plots
27 ftab=LINE(q,ft,0)
28 LABEL(fstat)='F-Statistic and Critical Value'
29 LABEL(ftab)='For Squared Coherence'
30 PLOT2(ftab,fstat,q,q,1,1,freq,0,.5,0,5)

```

Example 4.6

BIVARIATE AUTOREGRESSIVE MODELING

The macro **BAR.MAC** carries out the bivariate autoregressive model determination and cross-spectral estimation procedure described in Section 4.2.3.

```

1 ;;
2 ;;   BAR.MAC: macro to perform bivariate AR model fitting and
3 ;;           spectral estimation.
4 ;;
5 ;;   INPUT: x, y, n (data and sample size)
6 ;;           maxp (maximum AR order)
7 ;;           Q (number of frequencies between 0 and 1)
8 ;;
9 PAUSE
10 ;start
11 CORR2(x,y,n,maxp,1,r10,r20,rho120,rho1,rho2,rho12,rho21)
12 M=maxp
13 MACRO(corr2plt,start) ;form table and plots of correlations
14 ;
15 ;   Fit AR:
16 ;
17 ;ar2
18 A=CORRAR2(r10,r20,rho120,rho1,rho2,rho12,rho21,maxp,2,n,p,sigma,cat,ier)
19 LIST(p)
20 IF(p.eq.0,end)
21 LISTM(A,2,2,p)
22 LISTM(sigma,2,2)
23 ;
24 ;   AR Spectra:
25 ;
26 ARSP2(A,p,sigma,Q,f11,f22,f12r,f12i)
27 MACRO(crossp,start) ;this plots autospectra, coherence, and phase
28 ;end

```

Computational Problems

C4.1. Use the **FIG41** macro to generate a realization of length 60 from the process discussed in Section 4.1. Produce a scatterplot of $X_1(t)$ and $X_2(t+2)$ for $t = 1, \dots, 58$ and a scatterplot for $X_1(t)$ and $X_2(t-2)$ for $t = 3, \dots, 60$. Are these plots consistent with the discussion of Figure 4.1?

C4.2. Write a macro that will generate 100 realizations of length 100 from a bivariate process

$$X_2(t) = X_1(t) + .5X_1(t-1),$$

where $X_1 \sim \text{WN}(1)$, and for each realization calculate the periodograms of X_1 and X_2 (use the `CORR` command for each one). For a user-specified integer `nf`, store all 100 pairs of `nf`th values of the periodogram, and after all 100 pairs are generated, produce a scatterplot. Try out the macro for `nf=10` and `nf=40`. Are the results consistent with the fact that the coherence spectrum is identically one for this process?

C4.3. If X_1 and X_2 are independent $\text{AR}(2)$ processes with coefficients 0.4, 0.5, and 0.3, 0.9, respectively and error variances 1, approximate $\text{Var}(\hat{\rho}_{12}(1))$ for a realization of length 100. (Hint: See part (b) of Theorem 4.2.1.)

C4.4. Align the prewhitened gas furnace data so that the resulting cross-correlation is largest for lag 0 (write a macro that will find the cross-correlations for a specified shift), and then use the `CROSSPE` macro to find the estimate of the squared coherence. Does the alignment make any difference relative to Figure 4.7?

C4.5. The following macro will simulate a realization from a specified transfer function model:

```

1 ;;
2 ;;   SIMFILT.MAC: macro to simulate the transfer function model
3 ;;
4 ;;           X2(t) = b0*X1(t) + Sum(j=1,K) b(j)*X1(t-j) + eps(t)
5 ;;
6 ;;           where X1 is an AR(p) process with coefficients
7 ;;           alpha, and eps is WN(sig2)
8 ;;
9 ;;   INPUT: n (length of realization)
10 ;;          alpha, p, sig2 (variance of WN)
11 ;;          K, b0, b (array of length K with coefficients)
12 ;;          seed (random number generator seed)
13 ;;
14 ;;   OUTPUT: x1, x2 (arrays of length n containing X1 and X2)
15 ;;
16 PAUSE
17 ;start
18 n=n+K
19 x1=AR(1,alpha,p,1,seed,n,ier,r0)
20 x2=FILT(x1,b,b0,n,K)
21 kp1=K+1
22 x1=EXTRACT(x1,kp1,n)      ;pull off corresponding part of x1
23 n=n-K
24 e=WN(0,n)
25 x2=x2+(sig2*.5)*e

```

Use this macro to simulate a realization of length 100 when X_1 is the $\text{AR}(2)$ process with coefficients 0.3 and 0.9, and the filter coefficients are

$$\beta_0 = 1, \quad \beta_1 = 0.75, \quad \beta_2 = 0.50, \quad \beta_3 = 0.25.$$

Use the **FILTEST** macro to estimate these coefficients. Use $Q = 64$ and $Q = 128$ and see if it makes any difference.

C4.6. Compare the square modulus of the frequency transfer functions of the two sets of filter coefficients that we discussed for the gas furnace data.

Theoretical Problems

T4.1. Suppose that $\mathbf{X}(t) = (X_1(t), X_2(t))^T$ is a bivariate time series having autocovariance function \mathbf{R} and spectral density function \mathbf{f} . Define the univariate series Y by

$$Y(t) = h_1 X_1(t) + h_2 X_2(t).$$

a) Show that the autocovariance function of Y is given by

$$R(v) = h_1 R_{11}(v) + h_2 R_{22}(v) + h_1 h_2 R_{12}(v) + h_1 h_2 R_{21}(v).$$

b) Thus show that the spectral density function of Y is given by

$$f_Y(\omega) = \mathbf{h}^T \mathbf{f}(\omega) \mathbf{h},$$

where $\mathbf{h} = (h_1, h_2)^T$.

T4.2. If $\mathbf{X}(t) = (X_1(t), X_2(t))^T$ is a bivariate time series having spectral density \mathbf{f} and $Y_1(t) = g(L)X_1(t)$ and $Y_2(t) = h(L)X_2(t)$ are filtered versions of X_1 and X_2 respectively, then show that the coherency spectrum of $\mathbf{Y}(t) = (Y_1(t), Y_2(t))^T$ is the same as that for \mathbf{X} . (Hint: Use the bivariate Filter Theorem to find the auto- and cross-spectral densities for \mathbf{Y} .)

T4.3 Verify that the expression for \mathbf{f}_Y is correct in part (b) of Theorem 4.1.4, and show that the expressions for \mathbf{R}_Y and \mathbf{f}_Y reduce to those given in the univariate Filter Theorem if \mathbf{Y} and \mathbf{X} are univariate series.

T4.4. Suppose that Z_1 and Z_2 are two uncorrelated white noise series, each having variance 1, and we let

$$X_2(t) = \beta_1 Z_1(t) + \beta_2 Z_1(t-1) + Z_2(t), \quad X_1(t) = Z_1(t),$$

which we can also write as

$$X_2(t) = \beta_1 X_1(t) + \beta_2 X_1(t-1) + Z_2(t),$$

where X_1 and Z_2 are uncross-correlated white noise series. Show that

$$\begin{aligned} R_{12}(v) &= E[X_1(t) (\beta_1 X_1(t+v) + \beta_2 X_1(t+v-1) + Z_2(t+v))] \\ &= \beta_1 R_{11}(v) + \beta_2 R_{11}(v-1) \\ &= \begin{cases} \beta_1, & v = 0 \\ \beta_2, & v = 1 \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

and

$$f_{12}(\omega) = \beta_1 + \beta_2 e^{-2\pi i \omega} = \beta_1 + \beta_2 \cos 2\pi \omega - i\beta_2 \sin 2\pi \omega$$

and

$$c_{12}(\omega) = \beta_1 + \beta_2 \cos 2\pi \omega, \quad q_{12}(\omega) = \beta_2 \sin 2\pi \omega,$$

while

$$A_{12}(\omega) = \sqrt{\beta_1^2 + \beta_2^2 + 2\beta_1\beta_2 \cos 2\pi \omega}$$

and

$$\phi_{12}(\omega) = \arctan \left(\frac{-\beta_2 \sin 2\pi \omega}{\beta_1 + \beta_2 \cos 2\pi \omega} \right).$$

T4.5. What is the gain spectrum for the pure delay process discussed in Section 4.1.4?

T4.6. Show that

$$\int_0^1 \frac{f_{12}(\omega)}{f_{11}(\omega)} e^{2\pi i j \omega} d\omega = \int_0^1 \frac{c_{12}(\omega)}{f_{11}(\omega)} \cos 2\pi j \omega d\omega,$$

where c_{12} is the cospectral density of X_1 and X_2 .

T4.7. If X_1 is a white noise process with variance 1 and

$$X_2(t) = \frac{1}{2} \sum_{j=0}^{\infty} \left(\frac{1}{2} \right)^j X_1(t-j),$$

show that the cross-spectrum

$$f_{12}(\omega) = \frac{\frac{1}{2} (1 - \frac{1}{2} e^{2\pi i \omega})}{|1 - \frac{1}{2} e^{-2\pi i \omega}|^2}.$$

(Hint: Recall that if z is a complex number, then $1/z = \bar{z}/|z|^2$.)

T4.8. Express the coefficients of the fourth-degree polynomial

$$g(z) = |\mathbf{I}_2 + \mathbf{A}(1)z + \mathbf{A}(2)z^2|$$

in terms of the elements of the coefficient matrices $\mathbf{A}(1)$ and $\mathbf{A}(2)$ for a bivariate autoregressive process of order 2.

T4.9. Show that the bivariate sample spectral density can be written as

$$\hat{f}(\omega) = \frac{1}{n} \left(\sum_{t=1}^n (\mathbf{X}(t) - \bar{X}) e^{2\pi i t \omega} \right) \left(\sum_{s=1}^n (\mathbf{X}(s) - \bar{X}) e^{2\pi i s \omega} \right)^*.$$

T4.10. Show that a univariate AR(p) process X can be expressed as a first-order autoregressive process \mathbf{Y} of degree d ; that is, if $\mathbf{Y}^T(t) = (X(t), X(t-1), \dots, X(t-p+1))$ and $\boldsymbol{\epsilon}^T(t) = (\epsilon(t), 0, \dots, 0)$, then

$$\mathbf{Y}(t) + \mathbf{B}\mathbf{Y}(t-1) = \boldsymbol{\epsilon}(t),$$

where the $(p \times p)$ matrix \mathbf{B} (which is called the companion matrix) is given by

$$\mathbf{B} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_{p-1} & \alpha_p \\ -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & -1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 0 \end{bmatrix}.$$

T4.11. Suppose X is a periodic autoregressive process of period d , orders p_1, \dots, p_d , coefficients $\alpha_k(j)$, and residual variances $\sigma_1^2, \dots, \sigma_d^2$. Define the d -dimensional time series \mathbf{Y} by letting $\mathbf{Y}(1)$ consist of $X(1), \dots, X(d)$, $\mathbf{Y}(2)$ be the next d X 's, and so on. Show that \mathbf{Y} is a d -dimensional autoregressive process of order p , $(d \times d)$ coefficient matrices $\mathbf{A}(1), \dots, \mathbf{A}(p)$, and residual variance matrix $\boldsymbol{\Sigma}$, where

$$p = \left\lceil \frac{\max_{k=1, \dots, d} (p_k - k)}{d} \right\rceil + 1,$$

while $\boldsymbol{\Sigma} = \mathbf{L}^{-1} \mathbf{D} \mathbf{L}^{-T}$, $\mathbf{A}(j) = \mathbf{L}^{-1} \mathbf{B}(j)$, $\mathbf{D} = \text{Diag}(\sigma_1^2, \dots, \sigma_d^2)$, \mathbf{L} is a $(d \times d)$ unit lower triangular matrix having $\alpha_k(k-j)$ as its (k, j) th element, and $B_{kj}(v) = \alpha_k(dv - j + k)$, which is zero if $dv - j + k > p_k$.

APPENDIX A

Some Mathematical and Statistical Topics

In this appendix we discuss several ideas from mathematics and statistics that are used in the book. We begin with a description of some matrix operations.

A.1. Matrix Operations

In this section we consider three operations on matrices: (1) the modified Cholesky decomposition of a positive definite matrix, (2) the Gram-Schmidt decomposition of a nonsingular ($n \times m$) matrix, and (3) the matrix sweep operator. Each of these is useful as both a computational and theoretical tool.

A.1.1. The Modified Cholesky Decomposition

An ($n \times n$) matrix \mathbf{V} is said to be positive definite or positive semidefinite if for any nonzero n -dimensional vector \mathbf{h} , we have $\mathbf{h}^T \mathbf{V} \mathbf{h} > 0$ or $\mathbf{h}^T \mathbf{V} \mathbf{h} \geq 0$ respectively. In statistics it often necessary to find what is called the square root of a positive definite matrix.

Definition. The square root of a symmetric, positive definite ($n \times n$) matrix \mathbf{V} is an ($n \times n$) matrix \mathbf{A} satisfying $\mathbf{V} = \mathbf{A} \mathbf{A}^T$. We denote such a matrix by $\mathbf{V}^{1/2}$. The inverse square root of \mathbf{V} is the inverse of $\mathbf{V}^{1/2}$ and is denoted by $\mathbf{V}^{-1/2}$. The transposes of \mathbf{V}^{-1} and $\mathbf{V}^{-1/2}$ are denoted by \mathbf{V}^{-T} and $\mathbf{V}^{-T/2}$ respectively.

The next theorem shows that there exists a unique matrix square root for any positive definite matrix and also shows how to find it. Note that a triangular matrix is called unit if it has ones on the main diagonal.

Theorem A.1.1

MODIFIED CHOLESKY DECOMPOSITION

Let \mathbf{V} be a symmetric $(n \times n)$ matrix. Then

a) \mathbf{V} is positive definite if and only if there exists a unique unit lower triangular $(n \times n)$ matrix \mathbf{L} and a unique diagonal $(n \times n)$ matrix \mathbf{D} having positive diagonal elements, such that

$$\mathbf{V} = \mathbf{L}\mathbf{D}\mathbf{L}^T.$$

b) This factorization is called the modified Cholesky decomposition (MCD) of \mathbf{V} , and if the decomposition exists we can calculate the elements of \mathbf{L} and \mathbf{D} one row at a time by $D_{11} = V_{11}$ and for $i = 2, \dots, n$:

$$L_{ij} = \frac{V_{ij} - \sum_{l=1}^{j-1} L_{il} D_{ll} L_{jl}}{D_{jj}}, \quad j = 1, \dots, i-1$$

$$D_{ii} = V_{ii} - \sum_{l=1}^{i-1} D_{ll} L_{il}^2.$$

c) The MCD of \mathbf{V} is nested; that is, if \mathbf{V}_k , \mathbf{L}_k , and \mathbf{D}_k represent the upper left-hand $(k \times k)$ parts of \mathbf{V} , \mathbf{L} , and \mathbf{D} , respectively, then

$$\mathbf{V}_k = \mathbf{L}_k \mathbf{D}_k \mathbf{L}_k^T, \quad k = 1, \dots, n,$$

and thus for any k greater than or equal to i and j , the (i, j) th elements of \mathbf{V}_k , \mathbf{L}_k , and \mathbf{D}_k can be denoted by V_{ij} , L_{ij} , and D_{ij} , respectively.

d) The unique square root of a positive definite matrix \mathbf{V} is given by $\mathbf{V}^{1/2} = \mathbf{L}\mathbf{D}^{1/2}$ where $\mathbf{D}^{1/2} = \text{Diag}(D_{11}^{1/2}, \dots, D_{nn}^{1/2})$.

e) The inverse square root of a positive definite matrix \mathbf{V} is given by $\mathbf{V}^{-1/2} = \mathbf{D}^{-1/2} \mathbf{L}^{-1}$ where $\mathbf{D}^{-1/2} = \text{Diag}(D_{11}^{-1/2}, \dots, D_{nn}^{-1/2})$.

The MCHOL Command

If the array \mathbf{A} contains the symmetric $(p \times p)$ matrix \mathbf{A} , then the command

MCHOL(A,p,L,D,ier)

will attempt to find the Cholesky factors \mathbf{L} and \mathbf{D} of \mathbf{A} . If it judges that \mathbf{A} is not positive definite, that is, if a diagonal element of \mathbf{D} is found that is less

than 10^{-25} , it will return a 1 in the output integer variable **ier**. If **A** is judged to be positive definite, **MCHOL** will return a 0 in **ier**, and will return the factors **L** and **D** in the arrays **L** and **D**. Thus in addition to finding the decomposition if possible, **MCHOL** provides an easy check for positive definiteness of a matrix.

A.1.2. The Gram-Schmidt Decomposition

An important operation in many areas of mathematics and statistics is to find what is called an orthogonal basis for a matrix; that is, from one set of vectors, find a new set by some linear transformation such that the inner product of any two different new vectors is zero. We will use the Gram-Schmidt decomposition (see Clayton (1971)) to accomplish this aim.

Theorem A.1.2 GRAM-SCHMIDT DECOMPOSITION

If **X** is an $(n \times p)$ matrix having full rank p , then there exists an $(n \times p)$ orthogonal matrix **Q** (that is, $\mathbf{Q}^T \mathbf{Q}$ is diagonal) and a unit upper triangular $(p \times p)$ matrix **R** such that

$$\mathbf{X} = \mathbf{Q}\mathbf{R}.$$

This decomposition of **X** is called its Gram-Schmidt decomposition (GSD).

Note that $\mathbf{X}^T \mathbf{X} = \mathbf{R}^T \mathbf{Q}^T \mathbf{Q} \mathbf{R} = \mathbf{R}^T \mathbf{V} \mathbf{R}$ where \mathbf{R}^T is unit lower triangular and **V** is diagonal with positive diagonal elements. Since the modified Cholesky decomposition $\mathbf{X}^T \mathbf{X} = \mathbf{L} \mathbf{D} \mathbf{L}^T$ of $\mathbf{X}^T \mathbf{X}$ is unique, we have that $\mathbf{R}^T = \mathbf{L}$ and $\mathbf{V} = \mathbf{D}$.

The GS Command

If the array **X** contains the $(n \times p)$ matrix **X**, then the command

GS(X,n,p,Q,R,ier)

will attempt to find factors **Q** and **R** such that $\mathbf{X} = \mathbf{Q}\mathbf{R}$ by what is called the modified Gram-Schmidt decomposition algorithm; that is, if we let $\mathbf{Q}_0 = \mathbf{X}$, and $\mathbf{Q}_1, \dots, \mathbf{Q}_p = \mathbf{Q}$ be a sequence of $(n \times p)$ matrices, and \mathbf{q}_{ij} denote the j th column of \mathbf{Q}_i , then **GS** calculates for $i = 1, \dots, p-1$:

$$d_i = \mathbf{q}_{i-1,i}^T \mathbf{q}_{i-1,i}$$

$$R_{ij} = \frac{\mathbf{q}_{i-1,j}^T \mathbf{q}_{i-1,i}}{d_i}, \quad j = i+1, \dots, p$$

$$\mathbf{q}_{ij} = \mathbf{q}_{i-1,j} - R_{ij} \mathbf{q}_{i-1,i}, \quad j = i+1, \dots, p.$$

If $|d_i| < 10^{-25}$ for any i , then GS concludes that \mathbf{X} is singular and returns a 1 in the output integer variable **ier**. Otherwise, **ier** is returned as 0 and \mathbf{Q} and \mathbf{R} are returned in the arrays \mathbf{Q} and \mathbf{R} of length $\mathbf{n}*\mathbf{p}$ and $\mathbf{p}*\mathbf{p}$. Note that the algorithm is done in place; that is, $\mathbf{Q}_1, \dots, \mathbf{Q}_p$ are actually calculated in a single matrix. See Problem C1.25 for an illustration of the relationship between MCHOL and GS.

A.1.3. The Sweep Operator

Many of the computational and theoretical results in regression, multivariate analysis, and time series can be expressed succinctly using what is called the sweep operator.

Definition. Let \mathbf{A} be an $(n \times n)$ matrix. The process of sweeping \mathbf{A} on its k th diagonal element, denoted by $\mathbf{B} = \text{SWEEP}(k)\mathbf{A}$, is the process of forming the matrix \mathbf{B} by (assuming $A_{kk} \neq 0$):

$$\begin{aligned} B_{kk} &= \frac{1}{A_{kk}} \\ B_{ik} &= -\frac{A_{ik}}{A_{kk}}, \quad i \neq k \quad (k\text{th column}) \\ B_{kj} &= \frac{A_{kj}}{A_{kk}}, \quad j \neq k \quad (j\text{th row}) \\ B_{ij} &= A_{ij} - \frac{A_{ik}A_{kj}}{A_{kk}}, \quad i \neq k, j \neq k. \end{aligned}$$

If $\mathbf{B}_1 = \text{SWEEP}(k_1)\mathbf{A}$, $\mathbf{B}_2 = \text{SWEEP}(k_2)\mathbf{B}_1$, \dots , $\mathbf{B} = \text{SWEEP}(k_r)\mathbf{B}_{r-1}$, we write $\mathbf{B} = \text{SWEEP}(k_1, \dots, k_r)\mathbf{A}$, and say that \mathbf{B} is the result of sweeping \mathbf{A} on its diagonals k_1, \dots, k_r .

The sweep operator has many useful properties as we see in the next theorem.

Theorem A.1.3	PROPERTIES OF THE SWEEP OPERATOR
----------------------	----------------------------------

Let

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix},$$

where \mathbf{B} , \mathbf{C} , \mathbf{D} , and \mathbf{E} are $(r \times r)$, $(r \times s)$, $(s \times r)$, and $(s \times s)$ respectively. Then

a) $\text{SWEEP}(k_1, \dots, k_p)\mathbf{A}$ can be found by sweeping the diagonals in any order, for example,

$$\text{SWEEP}(k_1, k_2)\mathbf{A} = \text{SWEEP}(k_2, k_1)\mathbf{A}.$$

b) $\text{SWEEP}(k, k)\mathbf{A} = \mathbf{A}$; that is, sweeping a second time on a given diagonal undoes the effect of a previous sweeping on that diagonal.

$$\text{c) } \text{SWEEP}(1, \dots, r)\mathbf{A} = \begin{bmatrix} \mathbf{B}^{-1} & \mathbf{B}^{-1}\mathbf{C} \\ -\mathbf{D}\mathbf{B}^{-1} & \mathbf{E} - \mathbf{D}\mathbf{B}^{-1}\mathbf{C} \end{bmatrix} \text{ if } \mathbf{B} \text{ is nonsingular.}$$

$$\text{d) } \text{SWEEP}(r+1, \dots, r+s)\mathbf{A} = \begin{bmatrix} \mathbf{B} - \mathbf{C}\mathbf{E}^{-1}\mathbf{D} & -\mathbf{C}\mathbf{E}^{-1} \\ \mathbf{E}^{-1}\mathbf{D} & \mathbf{E}^{-1} \end{bmatrix} \text{ if } \mathbf{E} \text{ is nonsingular.}$$

$$\text{e) } \text{SWEEP}(1, \dots, r+s)\mathbf{A} = \mathbf{A}^{-1}$$

$$= \begin{bmatrix} \mathbf{B}^{-1} + \mathbf{B}^{-1}\mathbf{C}(\mathbf{E} - \mathbf{D}\mathbf{B}^{-1}\mathbf{C})^{-1}\mathbf{D}\mathbf{B}^{-1} & -\mathbf{B}^{-1}\mathbf{C}(\mathbf{E} - \mathbf{D}\mathbf{B}^{-1}\mathbf{C})^{-1} \\ -(\mathbf{E} - \mathbf{D}\mathbf{B}^{-1}\mathbf{C})^{-1}\mathbf{D}\mathbf{B}^{-1} & (\mathbf{E} - \mathbf{D}\mathbf{B}^{-1}\mathbf{C})^{-1} \end{bmatrix}$$

if \mathbf{B} , \mathbf{E} , and $(\mathbf{E} - \mathbf{D}\mathbf{B}^{-1}\mathbf{C})^{-1}$ are nonsingular.

The sweep operator can be used to solve many theoretical and computational problems. We consider two examples of its use. First, the basic quantities for the regression of \mathbf{y} on \mathbf{X} (where \mathbf{X} is $(n \times p)$) can be found by applying property (c) (see Section A.5):

$$\text{SWEEP}(1, \dots, p) \begin{bmatrix} \mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{y} \\ \mathbf{y}^T\mathbf{X} & \mathbf{y}^T\mathbf{y} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma^2} \text{Var}(\hat{\beta}) & \hat{\beta} \\ -\hat{\beta}^T & \text{RSS} \end{bmatrix}.$$

To illustrate the theoretical utility of sweep, notice for the matrix \mathbf{A} in Theorem A.1.3 that sweeping on all of its diagonals is the same as sweeping the matrix $\text{SWEEP}(r+1, \dots, r+s)\mathbf{A}$ on diagonals $1, \dots, r$. Thus the upper left-hand corner in part (e) is the inverse of the upper left-hand corner in part (d) by part (c). Doing a similar calculation with $\pm\mathbf{E}^{-1}$ replacing \mathbf{E} in \mathbf{A} gives the well-known matrix inversion lemma (see Rao (1973), p. 33 for example).

Theorem A.1.4	THE MATRIX INVERSION LEMMA
----------------------	-----------------------------------

Let \mathbf{B} and \mathbf{E} be nonsingular $(r \times r)$ and $(s \times s)$ matrices and \mathbf{C} and \mathbf{D} be $(r \times s)$ and $(s \times r)$ matrices. Then

$$(\mathbf{B} \pm \mathbf{CED})^{-1} = \mathbf{B}^{-1} \mp \mathbf{B}^{-1}\mathbf{C}(\mathbf{E}^{-1} \pm \mathbf{DB}^{-1}\mathbf{C})^{-1}\mathbf{DB}^{-1}.$$

We will use this formula extensively when we consider recursive regression in Section A.5, that is, when we consider adjusting estimators for the addition or deletion of observations. It can also be used to motivate the Kalman Filter Algorithm discussed in Section A.4.

The SWEEP Command

The SWEEP command has two forms. Suppose the array \mathbf{A} contains the $(n \times n)$ matrix \mathbf{A} . Then the command

```
B=SWEEP(A,n,k1,k2,ier)
```

attempts to form the matrix \mathbf{B} by sweeping \mathbf{A} on diagonals $k1$ through $k2$. The second form

```
B=SWEEP(A,n,ind,m,ier)
```

attempts to form \mathbf{B} by sweeping \mathbf{A} on the m diagonals whose indices are entered in the array \mathbf{ind} . If a diagonal element less than 10^{-25} is encountered, SWEEP will return a 1 in the output integer variable \mathbf{ier} . Otherwise, it will return a 0 in \mathbf{ier} and the matrix \mathbf{B} in the array \mathbf{B} of length $n*n$.

A.2. Fourier Series and Spectral Representations

The ideas of frequency domain analysis of data are an important part of time series analysis. We have attempted to motivate these ideas intuitively in the main part of this book. In this section we describe in more detail some of the mathematics behind the ideas. For more details the reader can consult Chapter 4 of Priestley (1981) or Chapter 7 of Anderson (1971). We recall that the basic idea of the frequency domain analysis of time series is to express various quantities as the "sum" of orthogonal sinusoids.

A.2.1. Fourier Series for Periodic Functions

Suppose that f is a function defined on a finite interval $[a, b]$. By this we also mean to include functions that are periodic of period $b - a$. Throughout this book we have taken our interval to be $[0, 1]$, as if it is not, we can construct $g(y) = f[(x-a)/(b-a)]$ and apply the results to the function g . This interval is also natural for spectral densities as then frequency can be thought of as cycles per unit of time. Thus for monthly data, frequency $1/12$ refers to possible cycles of length 12 months. The interval traditionally considered in mathematics is $[-\pi, \pi]$, which means for time series that one has to perform mental arithmetic involving π in order to convert frequency to a physically meaningful quantity. Using the interval $[0, 1]$ also simplifies many of the formulas in the subject.

Our aim is to decompose f into the sum of sinusoids that are independent in some sense. Thus we begin by trying to approximate f by a sum of sinusoids of periods $1, 1/2, \dots, 1/n$ for an arbitrary positive integer n .

Theorem A.2.1	FOURIER SERIES APPROXIMATION
----------------------	-------------------------------------

Let f be a square integrable function defined on $[0, 1]$; that is, $\int_0^1 f^2(x) dx < \infty$. Let

$$g_j(x) = a_j \cos 2\pi jx + b_j \sin 2\pi jx = C_j \cos(2\pi jx - \phi_j)$$

be a sinusoid of period $1/j$. Let

$$f_n(x) = \frac{a_0}{2} + \sum_{j=1}^n g_j(x),$$

where a_0 is some constant. Then the values of a_0, a_1, \dots, a_n and b_1, \dots, b_n that minimize

$$\int_0^1 \left| f(x) - \frac{a_0}{2} - \sum_{j=1}^n g_j(x) \right|^2 dx$$

are given by

$$a_j = 2 \int_0^1 \cos 2\pi jx f(x) dx, \quad j = 0, 1, \dots, n$$

$$b_j = 2 \int_0^1 \sin 2\pi jx f(x) dx, \quad j = 1, \dots, n.$$

Thus the sum of sinusoids that is closest to f in this integrated square error sense (which is analogous to the sum of squares of errors in regression

analysis) has the coefficients given in the theorem. Note that f need not be square integrable in order to calculate these coefficients. In fact, f need only be absolutely integrable, that is, $\int_0^1 |f(x)| dx < \infty$, since then for example

$$\begin{aligned} \int_0^1 |f(x) \cos 2\pi j x| dx &\leq \int_0^1 |f(x)| |\cos 2\pi j x| dx \\ &\leq \int_0^1 |f(x)| dx < \infty, \end{aligned}$$

since $|\cos 2\pi j x| \leq 1$. Note that a square integrable function is absolutely integrable but not necessarily vice versa (consider $f(x) = 1/\sqrt{x}$ for example).

Definition. Let f be an absolutely integrable function defined on $[0,1]$, and let

$$a_j = 2 \int_0^1 \cos 2\pi j x f(x) dx, \quad j = 0, 1, \dots$$

$$b_j = 2 \int_0^1 \sin 2\pi j x f(x) dx, \quad j = 1, 2, \dots$$

and

$$f_n(x) = \frac{a_0}{2} + \sum_{j=1}^n [a_j \cos 2\pi j x + b_j \sin 2\pi j x].$$

Then the a 's and b 's are called the cosine and sine Fourier coefficients of f , while f_n is called the n th partial sum Fourier approximant to f .

In general, we will have that $f \neq f_n$. To accomplish a harmonic analysis of f then, we hope that $f_n \rightarrow f$ in some sense, and also that there is some sense in which the sinusoids g_1, g_2, \dots are orthogonal. In the next theorem we present results showing that in many cases we do have the convergence and orthogonality.

Theorem A.2.2 PROPERTIES OF FOURIER SERIES

Let f be an absolutely integrable function defined on the interval $[0,1]$, and let f_1, f_2, \dots be its approximating Fourier sums. Then

a) The sinusoids $g_j(x) = a_j \cos 2\pi j x + b_j \sin 2\pi j x$ are orthogonal in the sense that

$$\int_0^1 g_j(x) g_k(x) dx = \begin{cases} 0, & j \neq k \\ a_j^2 + b_j^2, & j = k. \end{cases}$$

b) We have the following results about pointwise convergence of $f_n(x)$:

i) At a point x where f is continuous and the left and right derivatives of f exist,

$$\lim_{n \rightarrow \infty} f_n(x) = f(x).$$

ii) At a point x where f is discontinuous but the left and right derivatives of f exist,

$$\lim_{n \rightarrow \infty} f_n(x) = \frac{f(x^+) + f(x^-)}{2},$$

where $f(x^+)$ and $f(x^-)$ denote the right and left limits of f at the point x ; that is, the approximating Fourier sums converge to the average of the two values of f at the discontinuity.

iii) If f is continuous and has square integrable derivative, then $f_n \rightarrow f$ absolutely and uniformly.

c) If f is square integrable, then

$$\lim_{n \rightarrow \infty} \int_0^1 |f(x) - f_n(x)|^2 dx = 0,$$

while

$$2 \int_0^1 f^2(x) dx = \frac{a_0^2}{2} + \sum_{k=1}^{\infty} (a_k^2 + b_k^2).$$

Implications: This theorem gives us our harmonic analysis for two types of functions—absolutely integrable (part (b)) and square integrable (part (c)). For square integrable functions, $f_n \rightarrow f$ in the integrated squared error sense, which need not mean that $f_n(x)$ converges pointwise to $f(x)$. Note that the second equation in part (c) (which is called Parseval's identity) is analogous to a decomposition of "total sum of squares" in analysis of variance (the integral plays the role of the sum) into a sum of squares of amplitudes of the sinusoids for periods $1, 1/2, 1/3, \dots$. For absolutely integrable functions, part (b,i) shows that under general conditions, $f_n(x)$ does converge to $f(x)$ pointwise, while part (b,ii) explains the behavior of the Fourier approximants for the ideal bandpass filter example in Section 2.3. Thus at the two points of discontinuity in the transfer function, $f_n(x)$ converges to the average of 0 and 1, that is, to .5. Finally, part (a) shows that as long as the Fourier coefficients exist, then the sinusoids are orthogonal where now the inner product is defined as an integral rather than as a sum as it was in Section 1.4.3.

The Complex Form of Fourier Series

Since $e^{i\theta} = \cos \theta + i \sin \theta$, we have

$$\cos 2\pi jx = \frac{e^{2\pi i jx} + e^{-2\pi i jx}}{2} \quad \text{and} \quad \sin 2\pi jx = \frac{(-i)(e^{2\pi i jx} - e^{-2\pi i jx})}{2},$$

and thus

$$\begin{aligned} f_n(x) &= \frac{a_0}{2} + \sum_{j=1}^n [a_j \cos 2\pi jx + b_j \sin 2\pi jx] \\ &= \sum_{j=-n}^n r_j e^{-2\pi i jx}, \end{aligned}$$

where

$$r_j = \frac{a_j + ib_j}{2} \quad \text{and} \quad r_{-j} = \bar{r}_j = \frac{a_j - ib_j}{2}, \quad j = 0, 1, \dots$$

Note that

$$r_j = \int_0^1 f(x) e^{2\pi i jx} dx.$$

Thus we have that f and the r 's are Fourier pairs:

$$\begin{aligned} f(x) &= \sum_{j=-\infty}^{\infty} r_j e^{-2\pi i jx} \\ r_j &= \int_0^1 f(x) e^{2\pi i jx} dx. \end{aligned}$$

The r_j 's are called the complex Fourier coefficients of f .

A.2.2. Spectral Representation of R

In Section A.2.1 we were concerned with whether a function could be represented as a sum of sinusoids. In time series we have the converse problem; that is, we begin with the autocovariance sequence and ask whether there is a function having the R 's as its Fourier coefficients. The simplest, although not the most general, answer to this question is given in the following theorem.

Theorem A.2.3

SPECTRAL REPRESENTATION OF R

If $\{R(v), v \in \mathcal{Z}\}$ is a positive definite sequence of numbers such that $R(v) = R(-v)$ and $\sum_{v=-\infty}^{\infty} |R(v)| < \infty$, then there exists a continuous function f such that

$$a) f(\omega) = \sum_{v=-\infty}^{\infty} R(v) e^{-2\pi i v \omega}, \quad \omega \in [0, 1].$$

$$b) R(v) = \int_0^1 f(\omega) e^{2\pi i v \omega} d\omega, \quad v \in \mathcal{Z}.$$

$$c) f(\omega) = f(1 - \omega).$$

This theorem gives a harmonic analysis of an absolutely summable autocovariance function in terms of the spectral density function f . Unfortunately, the covariance function is not always absolutely summable, as illustrated by the harmonic process in Section 2.5.2. We can still obtain a harmonic analysis of a non-absolutely summable R as an integral, but the integral is of a special type called a Lebesgue-Stieltjes integral. In the spectral representation of R in Theorem 2.2.1, we wrote

$$R(v) = \int_0^1 \cos 2\pi v \omega dF(\omega),$$

where the integral is of the Lebesgue-Stieltjes type. Instead of defining this type of integral (which is rather complicated), we describe in the next theorem a way to calculate and interpret it in the special cases with which we are concerned.

Theorem A.2.4

LEBESGUE-STIELTJES INTEGRAL

Let g and F be bounded functions defined on $[0, 1]$ and for a positive integer m , consider partitioning $[0, 1]$ into the 2^m intervals

$$\left[0, \frac{1}{2^m}\right], \left[\frac{1}{2^m}, \frac{2}{2^m}\right], \dots, \left[\frac{2^m - 1}{2^m}, 1\right].$$

Thus if we let $\omega_{m,j} = j/2^m$, for $j = 0, 1, \dots, 2^m$, the k th interval is from $\omega_{m,k-1}$ to $\omega_{m,k}$, for $k = 1, \dots, 2^m$. Now let

$$I_m = \sum_{k=1}^{2^m} g(\omega'_{m,k}) [F(\omega_{m,k}) - F(\omega_{m,k-1})],$$

where $\omega'_{m,k} = (\omega_{m,k-1} + \omega_{m,k})/2$. Then if g is Lebesgue-Stieltjes integrable with respect to F , and g is continuous (and therefore is also Riemann-Stieltjes integrable), we have that $\lim_{m \rightarrow \infty} I_m$ exists and is finite, and

$$\int_0^1 g(x) dF(x) = \lim_{m \rightarrow \infty} I_m.$$

From this result we have that

$$R(v) = \int_0^1 \cos 2\pi v \omega dF(\omega) = \lim_{m \rightarrow \infty} \sum_{k=1}^{2^m} \cos 2\pi v \omega'_{m,k} [F(\omega_{m,k}) - F(\omega_{m,k-1})],$$

which means that the autocovariance sequence R can be regarded as the sum of a large number of cosines with amplitudes given by “increments” of the spectral distribution function F . Thus if there is a large increase in F between frequencies $\omega_{m,k-1}$ and $\omega_{m,k}$, the sinusoid having frequency $\omega'_{m,k}$ will be prominent in the decomposition of R . In fact, if the only large increase in F is in the interval containing $\omega'_{m,k}$, then R will appear almost exactly sinusoidal. The logical limit of this idea is the autocovariance of the harmonic process.

The Harmonic Process

Consider the harmonic process having a single frequency component of frequency $\omega_0 \in (0, .5)$; that is, $R(v) = \sigma^2 \cos 2\pi v \omega_0$. Define

$$F(\omega) = \begin{cases} 0, & 0 \leq \omega < \omega_0 \\ \frac{\sigma^2}{2}, & \omega_0 \leq \omega < 1 - \omega_0 \\ \sigma^2, & 1 - \omega_0 \leq \omega \leq 1, \end{cases}$$

and let ω_m be the center of the interval in the partition of $[0,1]$ at the m th step of the limiting process. If ω_0 is on the boundary between two intervals, we choose ω_m to be the center of the one containing the jump in F . Thus note that $1 - \omega_m$ is the center of the interval containing $1 - \omega_0$, and we have

$$\begin{aligned} \lim_{m \rightarrow \infty} \sum_{k=1}^{2^m} \cos 2\pi v \omega'_{m,k} [F(\omega_{m,k}) - F(\omega_{m,k-1})] \\ = \lim_{m \rightarrow \infty} \left[\frac{\sigma^2}{2} \cos 2\pi v \omega_m + \frac{\sigma^2}{2} \cos 2\pi v (1 - \omega_m) \right] \\ = \sigma^2 \cos 2\pi v \omega_0, \end{aligned}$$

since $\omega_m \rightarrow \omega_0$ and $\cos 2\pi v\omega = \cos 2\pi v(1 - \omega)$. Thus this definition of the spectral distribution function does indeed lead to the correct autocovariance function. We note that in the general theory of representing sequences, the function F is unique, and thus our definition of F is the only one that represents R .

In general, if F is absolutely continuous, we can see that the Lebesgue-Stieltjes integral reduces to the usual Riemann integral since then

$$\begin{aligned} \int_0^1 \cos 2\pi v\omega \, dF(\omega) &= \lim_{m \rightarrow \infty} \frac{1}{2^m} \sum_{k=1}^{2^m} \cos 2\pi v\omega'_{m,k} \left[\frac{F(\omega_{m,k}) - F(\omega_{m,k-1})}{1/2^m} \right] \\ &\doteq \lim_{m \rightarrow \infty} \frac{1}{2^m} \sum_{k=1}^{2^m} \cos 2\pi v\omega'_{m,k} \frac{dF(\omega'_{m,k})}{d\omega} \\ &= \int_0^1 \cos 2\pi v\omega f(\omega) \, d\omega \end{aligned}$$

by the definition of the Riemann integral.

A.2.3. Spectral Representation of X

It is possible to obtain a harmonic analysis of the time series X itself (see part (d) of Theorem 2.2.1). The representation is different than that for R in three ways. First, the representation for R is in terms of cosines only. This is because R is an even function, that is, $R(v) = R(-v)$. Second, the increments $F(\omega_k) - F(\omega_j)$ are replaced by the increments of two uncorrelated stochastic processes (see below). Finally, since the representation will be as a limit of sums of linear combinations of random variables, the usual limit cannot be used, rather the limit is a limit in mean square (see Section A.4).

A stochastic process $\{C(\omega), \omega \in [0, 1]\}$ is an indexed collection of random variables. The random variable $C(\omega_2) - C(\omega_1)$ for $0 \leq \omega_1 < \omega_2 \leq 1$ is called an increment of C . If increments defined on nonoverlapping intervals are uncorrelated, we say that C has uncorrelated increments.

Thus the spectral representation of X means essentially that

$$\begin{aligned} X(t) = \text{l.i.m.} \sum_{k=1}^{2^m} &\left[\cos 2\pi v\omega'_{m,k} [C(\omega_{m,k}) - C(\omega_{m,k-1})] \right. \\ &\left. + \sin 2\pi v\omega'_{m,k} [S(\omega_{m,k}) - S(\omega_{m,k-1})] \right], \end{aligned}$$

where l.i.m. denotes limit in mean square. This again allows us to think of a particular infinitely long realization from X as the sum of sinusoids whose amplitudes have been chosen according to some random mechanism (determined

by the processes C and S) whose properties (such as variance) are described by the spectral distribution function F .

A.3. The Fast Fourier Transform

Many of the calculations in time series analysis require the calculation of the discrete Fourier transform of a set of numbers (see Section 1.4.3 for example). In this section we describe the fast Fourier transform. The basic ideas were developed and popularized by Cooley and Tukey (1965) and Gentleman and Sande (1966). The FFT is a classic example of the use of recursion to greatly reduce the number of operations required to solve a computational problem. Although there are a wide variety of algorithms known as fast Fourier transforms, almost all of them have two essential elements: (1) a recursion that allows one to find the DFT of n points as a simple combination of DFT's for subsets of the points, and (2) a reordering of the original data so as to minimize the number of complex exponentials that need to be calculated and/or stored while the recursion is being performed.

A.3.1. The FFT Recursion

Let $W_n = e^{-2\pi i/n}$. Then we can write the DFT of $X(1), \dots, X(n)$ as

$$Z(k) = \sum_{t=1}^n X(t) W_n^{(t-1)(k-1)}, \quad k = 1, \dots, n.$$

Theorem A.3.1 THE FFT RECURSION

Suppose n can be factored into $n = rs$. Define the $(r \times s)$ matrix \mathbf{V} to have j th column

$$X(j), X(s+j), \dots, X((r-1)s+j), \quad j = 1, \dots, s.$$

Then

$$Z(k) = \sum_{l=1}^s W_n^{(k-1)(l-1)} Z_l(\text{mod}(k-1, r) + 1),$$

where $Z_l(1), \dots, Z_l(r)$ is the DFT of the l th column of \mathbf{V} .

Proof: We can get $Z(k)$ by adding over the elements of \mathbf{V} :

$$\begin{aligned} Z(k) &= \sum_{l=1}^s \sum_{m=1}^r V_{ml} W_n^{(k-1)[(m-1)s+(l-1)]} \\ &= \sum_{l=1}^s W_n^{(k-1)(l-1)} \sum_{m=1}^r V_{ml} W_n^{(k-1)(m-1)s} \end{aligned}$$

since $V_{ml} = X((m-1)s + l)$. But $s/n = 1/r$ which gives $W_n^s = W_r$. Further,

$$k-1 = \left[\frac{k-1}{r}\right]r + (k-1) - \left[\frac{k-1}{r}\right]r = \left[\frac{k-1}{r}\right]r + \text{mod}(k-1, r),$$

and thus

$$\begin{aligned} W_n^{(k-1)(m-1)s} &= W_r^{(m-1)\{[\frac{k-1}{r}]r + \text{mod}(k-1, r)\}} \\ &= W_r^{(m-1)\text{mod}(k-1, r)}, \end{aligned}$$

since $W_r^{[\frac{k-1}{r}]r} = 1$ because $e^{-2\pi i} = 1$. Thus

$$\begin{aligned} Z(k) &= \sum_{l=1}^s W_n^{(k-1)(l-1)} \sum_{m=1}^r V_{ml} W_r^{(m-1)\text{mod}(k-1, r)} \\ &= \sum_{l=1}^s W_n^{(k-1)(l-1)} Z_l(\text{mod}(k-1, r) + 1). \end{aligned}$$

Thus it takes $sr^2 = nr$ operations to calculate the DFT's of all s columns of \mathbf{V} and then ns operations to calculate all n Z 's, making a total of $nr + ns = n(r + s)$ operations instead of n^2 . Further, if r is itself factorable, we can use the same idea to more rapidly find the DFT's of the columns of \mathbf{V} . This process can be continued until all of the prime factors of n are exhausted, giving a total of $n(p_1 + \dots + p_K)$ operations where p_1, \dots, p_K are the prime factors of n .

A.3.2. Reordering Data

The recursion given in Theorem A.3.1 can be further improved if we can order the calculations in such a way that all of the ones using a certain complex exponential can be done at the same time. To illustrate this, consider the case of $n=8$ and let $s=2$:

$$Z(1) = Z_1(1) + W_8^0 Z_2(1)$$

$$Z(2) = Z_1(2) + W_8^1 Z_2(2)$$

$$Z(3) = Z_1(3) + W_8^2 Z_2(3)$$

$$Z(4) = Z_1(4) + W_8^3 Z_2(4)$$

$$Z(5) = Z_1(1) + W_8^4 Z_2(1)$$

$$Z(6) = Z_1(2) + W_8^5 Z_2(2)$$

$$Z(7) = Z_1(3) + W_8^6 Z_2(3)$$

$$Z(8) = Z_1(4) + W_8^7 Z_2(4)$$

where Z_1 is the DFT of the odd numbered X 's and Z_2 is the DFT of the even numbered X 's. Thus

$$\begin{aligned} Z_1(1) &= Z_{11}(1) + W_4^0 Z_{12}(1) & Z_2(1) &= Z_{21}(1) + W_4^0 Z_{22}(1) \\ Z_1(2) &= Z_{12}(2) + W_4^1 Z_{12}(2) & Z_2(2) &= Z_{21}(2) + W_4^1 Z_{22}(2) \\ Z_1(3) &= Z_{11}(1) + W_4^2 Z_{12}(1) & Z_2(3) &= Z_{21}(1) + W_4^2 Z_{22}(1) \\ Z_1(4) &= Z_{12}(2) + W_4^3 Z_{12}(2) & Z_2(4) &= Z_{21}(2) + W_4^3 Z_{22}(2) \end{aligned}$$

where

$$\begin{aligned} Z_{11}(1) &= X(1) + W_2^0 X(5) \\ Z_{11}(2) &= X(1) + W_2^1 X(5) \\ Z_{12}(1) &= X(3) + W_2^0 X(7) \\ Z_{12}(2) &= X(3) + W_2^1 X(7) \\ Z_{21}(1) &= X(2) + W_2^0 X(6) \\ Z_{21}(2) &= X(2) + W_2^1 X(6) \\ Z_{22}(1) &= X(4) + W_2^0 X(8) \\ Z_{22}(2) &= X(4) + W_2^1 X(8). \end{aligned}$$

Thus if we start by permuting the X 's we can summarize the calculations by

$$\begin{bmatrix} X(1) \\ X(5) \\ X(3) \\ X(7) \\ X(2) \\ X(6) \\ X(4) \\ X(8) \end{bmatrix} \rightarrow \begin{bmatrix} Z_{11}(1) \\ Z_{12}(1) \\ Z_{21}(1) \\ Z_{22}(1) \\ Z_{11}(2) \\ Z_{12}(2) \\ Z_{21}(2) \\ Z_{22}(2) \end{bmatrix} \rightarrow \begin{bmatrix} Z_1(1) \\ Z_2(1) \\ Z_1(2) \\ Z_2(2) \\ Z_1(3) \\ Z_2(3) \\ Z_1(4) \\ Z_2(4) \end{bmatrix} \rightarrow \begin{bmatrix} Z(1) \\ Z(2) \\ Z(3) \\ Z(4) \\ Z(5) \\ Z(6) \\ Z(7) \\ Z(8) \end{bmatrix},$$

where the first arrow means first combine all four consecutive pairs using W_2^0 and then again using W_2^1 . The second arrow means combine the first two pairs using W_4^0 , the next two using W_4^1 , the next two using W_4^2 , and the last two using W_4^3 . In the last step we combine the first pair using W_8^0 , the next pair using W_8^1 , and so on. Thus at each step we are fixing the complex exponential and then doing all combinations using that value.

To determine the index k of $X(j)$ in the permuted array, one need only write down the binary representation of $j-1$, reverse the bits, and then add one.

For example, $X(5)$ is mapped to position 2 since $5-1=4=100 \rightarrow 001 \rightarrow 010=2$. Note that if $X(j)$ gets mapped to position k , then $X(k)$ gets mapped to position j .

There are many variations on this theme and further simplifications that can be made. In TIMESLAB we use the algorithm described by Singleton (1969) which is in some ways still the standard by which newer methods are judged. We chose this algorithm because it performs the transform in place and will work for a wide variety of different prime factors, although as we saw in Section 1.4 it does require that n not have any large prime factors.

A.4. Random Vectors and the Multivariate Normal Distribution

In elementary statistics we often avoid treating vectors of random variables since the variables are independent and can be treated separately. In time series analysis this is usually not the case.

A.4.1. Basic Definitions

We begin our discussion with a series of definitions.

1. A d -dimensional random vector \mathbf{X} is a vector of d random variables, that is, $\mathbf{X} = (X_1, \dots, X_d)^T$, where X_1, \dots, X_d are random variables.

2. The joint cumulative distribution function (cdf) $F_{\mathbf{X}}$ of the random vector \mathbf{X} is given by

$$F_{\mathbf{X}}(\mathbf{x}) = F_{\mathbf{X}}(x_1, \dots, x_d) = \Pr(X_1 \leq x_1, \dots, X_d \leq x_d).$$

3. If there exists a function $f_{\mathbf{X}}$ of d variables such that

$$F_{\mathbf{X}}(\mathbf{x}) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_d} f_{\mathbf{X}}(\mathbf{y}) d\mathbf{y},$$

then \mathbf{X} is said to be a continuous random vector and $f_{\mathbf{X}}$ is called the joint probability density function (pdf) of \mathbf{X} .

4. If \mathbf{Y} and \mathbf{X} are r -dimensional and d -dimensional random vectors, then $\mathbf{Z} = (\mathbf{Y}^T, \mathbf{X}^T)^T$ is an $(r+d)$ -dimensional random vector. If $f_{\mathbf{Z}}$, $f_{\mathbf{Y}}$, and $f_{\mathbf{X}}$ are the pdf's of \mathbf{Z} , \mathbf{Y} , and \mathbf{X} respectively, then the functions

$$f_{\mathbf{Y}|\mathbf{X}} = \frac{f_{\mathbf{Z}}}{f_{\mathbf{X}}} \quad \text{and} \quad f_{\mathbf{X}|\mathbf{Y}} = \frac{f_{\mathbf{Z}}}{f_{\mathbf{Y}}}$$

are called the conditional pdf's of \mathbf{Y} given \mathbf{X} and \mathbf{X} given \mathbf{Y} , respectively.

5. The mean vector $\boldsymbol{\mu}_X$ and covariance matrix $\boldsymbol{\Sigma}_X$ are given by

$$\boldsymbol{\mu}_X = E(\mathbf{X}) = \begin{bmatrix} E(X_1) \\ \vdots \\ E(X_d) \end{bmatrix} = \int \cdots \int \mathbf{x} f_X(\mathbf{x}) d\mathbf{x}$$

$$\boldsymbol{\Sigma}_X = \text{Var}(\mathbf{X}) = E[(\mathbf{X} - \boldsymbol{\mu}_X)(\mathbf{X} - \boldsymbol{\mu}_X)^T]$$

$$= \int \cdots \int (\mathbf{x} - \boldsymbol{\mu}_X)(\mathbf{x} - \boldsymbol{\mu}_X)^T f_X(\mathbf{x}) d\mathbf{x}$$

$$= \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \text{Cov}(X_1, X_3) & \cdots & \text{Cov}(X_1, X_d) \\ & \text{Var}(X_2) & \text{Cov}(X_2, X_3) & \cdots & \text{Cov}(X_2, X_d) \\ & & \text{Var}(X_3) & & \vdots \\ & & & \ddots & \vdots \\ & \text{Symmetric} & & & \text{Var}(X_d) \end{bmatrix}.$$

If $E(\mathbf{X}) = \boldsymbol{\mu}_X$ and $\text{Var}(\mathbf{X}) = \boldsymbol{\Sigma}_X$ we will write

$$\mathbf{X} \sim (\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X).$$

The covariance matrix $\boldsymbol{\Sigma}_{XY}$ of \mathbf{X} and \mathbf{Y} is the $(d \times r)$ matrix whose (j, k) th element is $\text{Cov}(X_j, Y_k)$. The conditional mean $\boldsymbol{\mu}_{Y|X}$ and variance $\boldsymbol{\Sigma}_{Y|X}$ of a random vector \mathbf{Y} given \mathbf{X} are defined in the same way as $\boldsymbol{\mu}_Y$ and $\boldsymbol{\Sigma}_Y$, using the conditional pdf of \mathbf{Y} given \mathbf{X} .

6. If we partition the $(r + d)$ -dimensional random vector \mathbf{Z} into the r -dimensional and d -dimensional random vectors \mathbf{Y} and \mathbf{X} consisting of the first r and last d elements of \mathbf{Z} , then we can partition $\boldsymbol{\mu}_Z$ and $\boldsymbol{\Sigma}_Z$ similarly as:

$$\boldsymbol{\mu}_Z = \begin{bmatrix} \boldsymbol{\mu}_Y \\ \boldsymbol{\mu}_X \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_Z = \begin{bmatrix} \boldsymbol{\Sigma}_{YY} & \boldsymbol{\Sigma}_{YX} \\ \boldsymbol{\Sigma}_{XY} & \boldsymbol{\Sigma}_{XX} \end{bmatrix},$$

where in fact,

$$\boldsymbol{\Sigma}_{YY} = \text{Var}(\mathbf{Y}), \quad \boldsymbol{\Sigma}_{XX} = \text{Var}(\mathbf{X}),$$

$$\boldsymbol{\Sigma}_{YX} = \text{Cov}(\mathbf{Y}, \mathbf{X}), \quad \boldsymbol{\Sigma}_{XY} = \text{Cov}(\mathbf{X}, \mathbf{Y}) = \boldsymbol{\Sigma}_{YX}^T.$$

7. The r -dimensional random vector \mathbf{Y} is called a linear transformation of the d -dimensional random vector \mathbf{X} if $\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{b}$ where \mathbf{A} is an $(r \times d)$ matrix of constants and \mathbf{b} is an r -dimensional vector of constants.

8. If \mathbf{X} is a d -dimensional random vector and \mathbf{A} is a $(d \times d)$ matrix of constants, then the scalar random variable $Q = \mathbf{X}^T \mathbf{A} \mathbf{X}$ is called a quadratic form in \mathbf{X} and \mathbf{A} is called the matrix of the quadratic form.

A.4.2. Basic Facts

With these definitions in mind, we next state some theoretical results that are used throughout the book. We begin with the following theorem.

Theorem A.4.1	MEAN AND VARIANCE OF A LINEAR FUNCTION
----------------------	---

If $\mathbf{X} \sim (\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$ then

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{b} \sim (\mathbf{A}\boldsymbol{\mu}_X + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}_X\mathbf{A}^T),$$

that is,

$$\mathbf{E}(\mathbf{A}\mathbf{X} + \mathbf{b}) = \mathbf{A}\mathbf{E}(\mathbf{X}) + \mathbf{b}$$

$$\text{Var}(\mathbf{A}\mathbf{X} + \mathbf{b}) = \mathbf{A}\text{Var}(\mathbf{X})\mathbf{A}^T.$$

Most of the theory and methods of time series analysis are built upon the properties of the multivariate normal distribution.

Definition. The d -dimensional random vector \mathbf{X} having mean $\boldsymbol{\mu}_X$ and variance $\boldsymbol{\Sigma}_X$ is said to have the d -dimensional normal distribution if its pdf is given by

$$f_X(\mathbf{x}) = (2\pi)^{-d/2} |\boldsymbol{\Sigma}_X|^{-1/2} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_X)^T \boldsymbol{\Sigma}_X^{-1} (\mathbf{x} - \boldsymbol{\mu}_X) \right),$$

for $\mathbf{x} \in \mathcal{R}^d$, that is, in d -dimensional Euclidean space. Such a random vector is denoted by $\mathbf{X} \sim N_d(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$.

The important properties of the multivariate normal distribution are summarized in Theorem A.4.2 (see Rao (1973), Chapter 8 for example).

Theorem A.4.2

PROPERTIES OF THE MULTIVARIATE NORMAL

If

$$\mathbf{Z} = \begin{pmatrix} \mathbf{Y} \\ \mathbf{X} \end{pmatrix} \sim N_{r+d} \left(\begin{bmatrix} \boldsymbol{\mu}_Y \\ \boldsymbol{\mu}_X \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{YY} & \boldsymbol{\Sigma}_{YX} \\ \boldsymbol{\Sigma}_{XY} & \boldsymbol{\Sigma}_{XX} \end{bmatrix} \right),$$

then

a) The characteristic function of \mathbf{Z} is given by

$$\phi_{\mathbf{Z}}(\mathbf{t}) = E(e^{i\mathbf{t}^T \mathbf{Z}}) = \exp(i\mathbf{t}^T \boldsymbol{\mu}_{\mathbf{Z}} - \frac{1}{2} \mathbf{t}^T \boldsymbol{\Sigma}_{\mathbf{Z}} \mathbf{t}).$$

b) If \mathbf{A} is an $s \times (r + d)$ matrix of constants and \mathbf{b} is an s -dimensional vector of constants, then

$$\mathbf{AZ} + \mathbf{b} \sim N_s(\mathbf{AZ} + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}_{\mathbf{Z}}\mathbf{A}^T);$$

that is, linear transformations of normal random vectors are also normally distributed.

c) i) The conditional distribution of \mathbf{Y} given \mathbf{X} is

$$\mathbf{Y}|\mathbf{X} \sim N_r(\boldsymbol{\mu}_Y + \boldsymbol{\Sigma}_{YX}\boldsymbol{\Sigma}_{XX}^{-1}(\mathbf{X} - \boldsymbol{\mu}_X), \boldsymbol{\Sigma}_{YY} - \boldsymbol{\Sigma}_{YX}\boldsymbol{\Sigma}_{XX}^{-1}\boldsymbol{\Sigma}_{XY});$$

that is, the pdf of \mathbf{Y} given \mathbf{X} is that of an r -dimensional normal random vector with mean and variance given by

$$\boldsymbol{\mu}_{Y|X} = E(\mathbf{Y}|\mathbf{X}) = \boldsymbol{\mu}_Y + \boldsymbol{\Sigma}_{YX}\boldsymbol{\Sigma}_{XX}^{-1}(\mathbf{X} - \boldsymbol{\mu}_X)$$

$$\boldsymbol{\Sigma}_{Y|X} = \text{Var}(\mathbf{Y}|\mathbf{X}) = \boldsymbol{\Sigma}_{YY} - \boldsymbol{\Sigma}_{YX}\boldsymbol{\Sigma}_{XX}^{-1}\boldsymbol{\Sigma}_{XY};$$

that is, the conditional mean of \mathbf{Y} given $\mathbf{X} = \mathbf{x}$ is a linear function of \mathbf{x} , while the conditional variance is the same no matter what \mathbf{x} is.

ii) Conversely, if

$$\mathbf{Y}|\mathbf{X} \sim N_r(\boldsymbol{\mu}_Y + \boldsymbol{\Sigma}_{YX}\boldsymbol{\Sigma}_{XX}^{-1}(\mathbf{X} - \boldsymbol{\mu}_X), \boldsymbol{\Sigma}_{YY} - \boldsymbol{\Sigma}_{YX}\boldsymbol{\Sigma}_{XX}^{-1}\boldsymbol{\Sigma}_{XY})$$

and $\mathbf{X} \sim N_r(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_{XX})$, then

$$\mathbf{Z} = \begin{pmatrix} \mathbf{Y} \\ \mathbf{X} \end{pmatrix} \sim N_{r+d} \left(\begin{bmatrix} \boldsymbol{\mu}_Y \\ \boldsymbol{\mu}_X \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{YY} & \boldsymbol{\Sigma}_{YX} \\ \boldsymbol{\Sigma}_{XY} & \boldsymbol{\Sigma}_{XX} \end{bmatrix} \right).$$

d) If \mathbf{A} is a $(d \times d)$ nonsingular matrix of constants, then the distribution of $\mathbf{Y}|\mathbf{X}$ and $\mathbf{Y}|\mathbf{AX}$ are the same.

e) $\text{Cov}(\mathbf{AX}, \mathbf{BY}) = \mathbf{A}\Sigma_{XY}\mathbf{B}^T.$

f) The quadratic form

$$Q = (\mathbf{X} - \boldsymbol{\mu}_X)^T \Sigma_{XX}^{-1} (\mathbf{X} - \boldsymbol{\mu}_X)$$

has the chi-square distribution with d degrees of freedom, and we write $Q \sim \chi_d^2$.

g) The quadratic form

$$Q_A = (\mathbf{X} - \boldsymbol{\mu}_X)^T \mathbf{A} (\mathbf{X} - \boldsymbol{\mu}_X)$$

has the same distribution as the random variable $\sum_{j=1}^s \lambda_j Z_j^2$, where Z_1, \dots, Z_s are independent χ_1^2 random variables, and the λ 's are the nonzero eigenvalues of $\mathbf{A}\Sigma_{XX}$. This reduces to the result of part (f) if $\mathbf{A} = \Sigma_{XX}^{-1}$ since the eigenvalues of the d -dimensional identity matrix are just one repeated d times, and the sum of d independent χ_1^2 random variables is χ_d^2 .

h) The mean and variance of a quadratic form in normal variables are given by

$$E(\mathbf{X}^T \mathbf{A} \mathbf{X}) = \text{tr}(\mathbf{A}\Sigma_{XX}) + \boldsymbol{\mu}_X^T \mathbf{A} \boldsymbol{\mu}_X$$

$$\text{Var}(\mathbf{X}^T \mathbf{A} \mathbf{X}) = 2\text{tr}[(\mathbf{A}\Sigma_{XX})^2] + 4\boldsymbol{\mu}_X^T \mathbf{A} \Sigma_{XX} \mathbf{A} \boldsymbol{\mu}_X.$$

i) The two quadratic forms $Q_1 = \mathbf{X}^T \mathbf{A} \mathbf{X}$ and $Q_2 = \mathbf{X}^T \mathbf{B} \mathbf{X}$ are independent if and only if $\mathbf{A}\Sigma_{XX}\mathbf{B}^T = \mathbf{0}$.

j) If Q_1 and Q_2 are two independent χ^2 random variables having n_1 and n_2 degrees of freedom, then the random variable

$$F = \frac{Q_1/n_1}{Q_2/n_2}$$

has the F distribution with n_1 and n_2 degrees of freedom and we write $F \sim F_{n_1, n_2}$.

k) If $Z \sim N_1(\mu, \sigma^2)$ and $Q \sim \chi_v^2$ are independent, then the random variable

$$t = \frac{(Z - \mu)/\sigma}{\sqrt{Q/v}}$$

has the Student- t distribution with v degrees of freedom and we write $t \sim t_v$.

The DIST Command

The pdf, cdf, and quantile functions for the $N(0,1)$, t_v , χ_v^2 , and F_{v_1, v_2} distributions can be evaluated via the DIST command, as can the general incomplete gamma function.

Simultaneous Confidence Intervals

A result that is useful in finding simultaneous confidence bands for functions is given in the next theorem (see Scheffé (1959), p. 406).

Theorem A.4.3	SIMULTANEOUS CONFIDENCE INTERVALS
----------------------	--

Let \mathbf{X} be an arbitrary p -dimensional random vector and let \mathbf{V} be a positive definite ($p \times p$) matrix of constants. Then the p -dimensional vector \mathbf{a} satisfies

$$(\mathbf{X} - \mathbf{a})^T \mathbf{V} (\mathbf{X} - \mathbf{a}) \leq 1$$

if and only if

$$|\mathbf{h}^T (\mathbf{X} - \mathbf{a})| \leq (\mathbf{h}^T \mathbf{V}^{-1} \mathbf{h})^{-1/2} = D$$

for all p -dimensional vectors \mathbf{h} ; that is,

$$\mathbf{h}^T \mathbf{X} - D \leq \mathbf{h}^T \mathbf{a} \leq \mathbf{h}^T \mathbf{X} + D.$$

A.4.3. Prediction of a Random Vector

Consider random vectors \mathbf{Y} and \mathbf{X} such that \mathbf{Y} has d elements, \mathbf{X} has r elements, and

$$\begin{aligned} E \begin{pmatrix} \mathbf{Y} \\ \mathbf{X} \end{pmatrix} &= \begin{bmatrix} \boldsymbol{\mu}_X \\ \boldsymbol{\mu}_Y \end{bmatrix} \\ \text{Var} \begin{pmatrix} \mathbf{Y} \\ \mathbf{X} \end{pmatrix} &= \begin{bmatrix} \boldsymbol{\Sigma}_{YY} & \boldsymbol{\Sigma}_{YX} \\ \boldsymbol{\Sigma}_{XY} & \boldsymbol{\Sigma}_{XX} \end{bmatrix}. \end{aligned}$$

We consider two criteria for predicting \mathbf{Y} from \mathbf{X} : (1) best unbiased prediction and (2) best linear unbiased prediction.

Definition. The best unbiased predictor of \mathbf{Y} from \mathbf{X} is that function $\tilde{\mathbf{Y}}$ of \mathbf{X} that has the same expectation as \mathbf{Y} and is closest to \mathbf{Y} in expected mean square error; that is,

$$E[(\mathbf{Y} - \mathbf{Y}^*)(\mathbf{Y} - \mathbf{Y}^*)^T] - E[(\mathbf{Y} - \tilde{\mathbf{Y}})(\mathbf{Y} - \tilde{\mathbf{Y}})^T]$$

is positive semidefinite for any other unbiased function \mathbf{Y}^* of \mathbf{X} . The best linear unbiased predictor $\hat{\mathbf{Y}}$ of \mathbf{Y} given \mathbf{X} is that linear function of \mathbf{X} having the same expectation as \mathbf{Y} and is closest to \mathbf{Y} in expected mean square error, that is, $\hat{\mathbf{Y}} = \hat{\mathbf{A}}\mathbf{X}$ where

$$E[(\mathbf{Y} - \mathbf{A}^*\mathbf{X})(\mathbf{Y} - \mathbf{A}^*\mathbf{X})^T] - E[(\mathbf{Y} - \hat{\mathbf{A}}\mathbf{X})(\mathbf{Y} - \hat{\mathbf{A}}\mathbf{X})^T]$$

is positive semidefinite for any other linear function $\mathbf{A}^*\mathbf{X}$ that is also unbiased for \mathbf{Y} .

It will turn out that $\tilde{\mathbf{Y}}$ is the conditional expectation of \mathbf{Y} given \mathbf{X} . Conditional expectation is a crucial concept in time series analysis and can be defined at many levels of generality. A more general and useful definition of conditional expectation than that given in Definition 5 in Section A.4.1 is given by the following (see Parzen (1962b), for example).

Definition. Let \mathbf{Y} be a random vector having finite mean. Then the conditional expectation $E(\mathbf{Y}|\mathbf{X})$ of \mathbf{Y} given \mathbf{X} is that function ϕ of \mathbf{X} such that

$$E([\mathbf{Y} - \phi(\mathbf{X})]\mathbf{g}(\mathbf{X})) = \mathbf{0}$$

for any bounded function \mathbf{g} of \mathbf{X} .

This definition is called descriptive (as opposed to the constructive definition in Section A.4.1) as it says that the “residual” $\mathbf{Y} - E(\mathbf{Y}|\mathbf{X})$ is uncorrelated with any other function of \mathbf{X} . In this sense there is no more information about \mathbf{Y} in \mathbf{X} after having removed $E(\mathbf{Y}|\mathbf{X})$. We note that the conditional variance $\text{Var}(\mathbf{Y}|\mathbf{X})$ of \mathbf{Y} given \mathbf{X} is just the conditional expectation of the random variable $[(\mathbf{Y} - E(\mathbf{Y}|\mathbf{X}))(\mathbf{Y} - E(\mathbf{Y}|\mathbf{X}))^T]$ given \mathbf{X} .

Letting $\mathbf{g}(\mathbf{X}) = \mathbf{1}$ in this definition yields $E(\phi(\mathbf{X})) = E(\mathbf{Y})$, and thus

$$\text{Cov}(\mathbf{Y} - \phi(\mathbf{X}), \mathbf{g}(\mathbf{X})) = \mathbf{0}.$$

With these definitions in mind, we have the following theorem. For simplicity we assume that the random vectors involved have zero mean.

Theorem A.4.4

GENERAL PREDICTION THEORY

Let \mathbf{Y} and \mathbf{X} be jointly distributed random vectors having zero mean. Then

a) The best unbiased predictor $\tilde{\mathbf{Y}}$ of \mathbf{Y} given \mathbf{X} and its error variance $\Sigma_{\tilde{\mathbf{Y}}} = \text{Var}(\mathbf{Y} - \tilde{\mathbf{Y}})$ are given by

$$\tilde{\mathbf{Y}} = \mathbf{E}(\mathbf{Y}|\mathbf{X}) \quad \text{and} \quad \Sigma_{\tilde{\mathbf{Y}}} = \text{Var}(\mathbf{Y}|\mathbf{X}).$$

b) If Σ_{XX} is nonsingular, then the best linear unbiased predictor $\hat{\mathbf{Y}}$ of \mathbf{Y} given \mathbf{X} and its error variance $\Sigma_{\hat{\mathbf{Y}}} = \text{Var}(\mathbf{Y} - \hat{\mathbf{Y}})$ are given by

$$\hat{\mathbf{Y}} = \hat{\mathbf{A}}\mathbf{X} \quad \text{and} \quad \Sigma_{\hat{\mathbf{Y}}} = \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY},$$

where $\hat{\mathbf{A}}$ satisfies the prediction normal equations

$$\Sigma_{XX}\hat{\mathbf{A}}^T = \Sigma_{XY}.$$

c) If \mathbf{Y} and \mathbf{X} are jointly multivariate normal, then $\hat{\mathbf{Y}} = \tilde{\mathbf{Y}}$ and $\Sigma_{\hat{\mathbf{Y}}} = \Sigma_{\tilde{\mathbf{Y}}}$.

Proof: To prove part (a), let \mathbf{Y}^* be an arbitrary function of \mathbf{X} having mean 0, and let

$$\boldsymbol{\delta} = \mathbf{Y} - \mathbf{E}(\mathbf{Y}|\mathbf{X}), \quad \boldsymbol{\gamma} = \mathbf{E}(\mathbf{Y}|\mathbf{X}) - \mathbf{Y}^*, \quad \boldsymbol{\alpha} = \mathbf{Y} - \mathbf{Y}^* = \boldsymbol{\delta} + \boldsymbol{\gamma}.$$

We need to show that $\mathbf{E}(\boldsymbol{\alpha}\boldsymbol{\alpha}^T) - \mathbf{E}(\boldsymbol{\delta}\boldsymbol{\delta}^T)$ is positive semidefinite. We have

$$\begin{aligned} \mathbf{E}(\boldsymbol{\alpha}\boldsymbol{\alpha}^T) - \mathbf{E}(\boldsymbol{\delta}\boldsymbol{\delta}^T) &= \mathbf{E}[(\boldsymbol{\delta} + \boldsymbol{\gamma})(\boldsymbol{\delta} + \boldsymbol{\gamma})^T] - \mathbf{E}(\boldsymbol{\delta}\boldsymbol{\delta}^T) \\ &= \mathbf{E}(\boldsymbol{\gamma}\boldsymbol{\delta}^T) + \mathbf{E}(\boldsymbol{\delta}\boldsymbol{\gamma}^T) + \mathbf{E}(\boldsymbol{\gamma}\boldsymbol{\gamma}^T), \end{aligned}$$

the third term of which is positive semidefinite since for any vector \mathbf{h} not identically zero, we have

$$\mathbf{h}^T \mathbf{E}(\boldsymbol{\gamma}\boldsymbol{\gamma}^T) \mathbf{h} = \mathbf{E}(\mathbf{h}^T \boldsymbol{\gamma}\boldsymbol{\gamma}^T \mathbf{h}) = \mathbf{E}[(\mathbf{h}^T \boldsymbol{\gamma})^2] \geq 0.$$

Now $\mathbf{E}(\boldsymbol{\gamma}) = \mathbf{E}(\boldsymbol{\delta}) = \mathbf{0}$ and thus $\mathbf{E}(\boldsymbol{\gamma}\boldsymbol{\delta}^T) = \text{Cov}(\boldsymbol{\gamma}, \boldsymbol{\delta})$ and $\mathbf{E}(\boldsymbol{\delta}\boldsymbol{\gamma}^T) = \text{Cov}(\boldsymbol{\delta}, \boldsymbol{\gamma})$. But both of these covariances are zero since $\boldsymbol{\gamma} = \mathbf{E}(\mathbf{Y}|\mathbf{X}) - \mathbf{Y}^*$ is a function of \mathbf{X} since $\mathbf{E}(\mathbf{Y}|\mathbf{X})$ and \mathbf{Y}^* are both functions of \mathbf{X} .

To prove part (b), let \mathbf{A} be an arbitrary $(r \times d)$ matrix. Then

$$\begin{aligned} S(\mathbf{A}) &= E[(\mathbf{Y} - \mathbf{A}\mathbf{X})(\mathbf{Y} - \mathbf{A}\mathbf{X})^T] \\ &= E(\mathbf{Y}\mathbf{Y}^T) - \mathbf{A}E(\mathbf{X}\mathbf{Y}^T) - E(\mathbf{Y}\mathbf{X}^T)\mathbf{A}^T + \mathbf{A}E(\mathbf{X}\mathbf{X}^T)\mathbf{A}^T \\ &= \Sigma_{YY} - \mathbf{A}\Sigma_{XY} - \Sigma_{YX}\mathbf{A}^T + \mathbf{A}\Sigma_{XX}\mathbf{A}^T \\ &= (\mathbf{A} - \Sigma_{YX}\Sigma_{XX}^{-1})\Sigma_{XX}(\mathbf{A} - \Sigma_{YX}\Sigma_{XX}^{-1})^T \\ &\quad + \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}. \end{aligned}$$

Thus $\Sigma_{\hat{Y}} = S(\hat{\mathbf{A}}) = \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}$ and

$$S(\mathbf{A}^*) - S(\hat{\mathbf{A}}) = (\mathbf{A}^* - \Sigma_{YX}\Sigma_{XX}^{-1})\Sigma_{XX}(\mathbf{A}^* - \Sigma_{YX}\Sigma_{XX}^{-1})^T$$

which is clearly positive semidefinite for any matrix \mathbf{A}^* .

Finally, part (c) follows from part (c) of Theorem A.4.2.

The Kalman Filter Algorithm

If we have a sequence of random vectors that are multivariate normal and satisfy a recursive relationship of a particular form, then predictors can also be found recursively. This type of recursion was used extensively by Kalman (1960) and thus the resulting algorithm is usually referred to as the Kalman Filter Algorithm.

Theorem A.4.5 THE KALMAN FILTER

Let $\mathbf{X}_0, \mathbf{X}_1, \dots$ and $\mathbf{Z}_1, \mathbf{Z}_2, \dots$ be sequences of r - and d -dimensional random vectors satisfying the two equations (called the state and observation equations respectively) for $t \geq 1$:

$$\mathbf{X}_t = \mathbf{A}_t\mathbf{X}_{t-1} + \mathbf{w}_t$$

$$\mathbf{Z}_t = \mathbf{H}_t\mathbf{X}_t + \mathbf{v}_t,$$

where the \mathbf{A}_t and \mathbf{H}_t are full-rank $(r \times r)$ and $(d \times r)$ matrices of constants,

$$\mathbf{w}_t \sim N_r(\mathbf{0}_r, \mathbf{W}_t)$$

$$\mathbf{v}_t \sim N_d(\mathbf{0}_d, \mathbf{V}_t),$$

all of the \mathbf{w}_t 's and \mathbf{v}_t 's are independent, and

$$\mathbf{X}_0 \sim N_r(\boldsymbol{\mu}_0, \boldsymbol{\Gamma}_0).$$

Let $\hat{\mathbf{X}}_t$ and $\hat{\Sigma}_t$ denote the mean and variance of the conditional distribution of \mathbf{X}_t given $\mathbf{Z}_1, \dots, \mathbf{Z}_t$. Then

a) For $t \geq 1$,

$$\hat{\mathbf{X}}_t = \mathbf{A}_t \hat{\mathbf{X}}_{t-1} + \mathbf{R}_t \mathbf{H}_t^T (\mathbf{H}_t \mathbf{R}_t \mathbf{H}_t^T + \mathbf{V}_t)^{-1} \mathbf{e}_t$$

$$\hat{\Sigma}_t = \mathbf{R}_t - \mathbf{R}_t \mathbf{H}_t^T (\mathbf{H}_t \mathbf{R}_t \mathbf{H}_t^T + \mathbf{V}_t)^{-1} \mathbf{H}_t \mathbf{R}_t,$$

where

$$\mathbf{R}_t = \mathbf{A}_t \hat{\Sigma}_{t-1} \mathbf{A}_t^T + \mathbf{W}_t$$

$$\mathbf{e}_t = \mathbf{Z}_t - \mathbf{H}_t \mathbf{A}_t \hat{\mathbf{X}}_{t-1},$$

and

$$\hat{\Sigma}_0 = \Sigma_0, \quad \hat{\mathbf{X}}_0 = \mu_0.$$

b) For $t > 1$ and $h \geq 0$, let $\tilde{\mathbf{X}}_{t+h-1|t-1}$ and $\tilde{\Sigma}_{t+h-1|t-1}$ denote the mean and variance of \mathbf{X}_{t+h-1} given $\mathbf{Z}_1, \dots, \mathbf{Z}_{t-1}$. Then

$$\tilde{\mathbf{X}}_{t+h-1|t-1} = \begin{cases} \hat{\mathbf{X}}_{t-1}, & h = 0 \\ \mathbf{A}_{t+h-1} \tilde{\mathbf{X}}_{t+h-2|t-1}, & h \geq 1 \end{cases}$$

$$\tilde{\Sigma}_{t+h-1|t-1} = \begin{cases} \hat{\Sigma}_{t-1}, & h = 0 \\ \mathbf{A}_{t+h-1} \tilde{\Sigma}_{t+h-2|t-1} \mathbf{A}_{t+h-1}^T + \mathbf{W}_{t+h-1}, & h \geq 1. \end{cases}$$

Proof: In our proof, we follow Meinhold and Singpurwalla (1983). Let \mathbf{Z}^t represent the set of variables $\{\mathbf{Z}_1, \dots, \mathbf{Z}_t\}$. We know that the distribution of $\mathbf{X}_t | \mathbf{Z}^t$, which is what we want, is normal since \mathbf{X}_0 and all of the \mathbf{w}_t 's and \mathbf{v}_t 's are normal, and thus all of the \mathbf{X}_t 's and \mathbf{Z}_t 's are jointly normal. We want to show that the mean and variance of $\mathbf{X}_t | \mathbf{Z}^t$ satisfy the relations in part (a). Note that $\hat{\mathbf{X}}_{t-1} = E(\mathbf{X}_{t-1} | \mathbf{Z}^{t-1})$ is a linear function of \mathbf{Z}^{t-1} and thus $\mathbf{e}_t = \mathbf{Z}_t - \mathbf{H}_t \mathbf{A}_t \hat{\mathbf{X}}_{t-1}$ and \mathbf{Z}^{t-1} are a linear transformation of \mathbf{Z}^t . Thus by part (d) of Theorem A.4.2, we know that the conditional distribution of $\mathbf{X}_t | \mathbf{Z}^t$ is the same as that of $\mathbf{X}_t | \mathbf{e}_t, \mathbf{Z}^{t-1}$. The distribution of $\mathbf{X}_t | \mathbf{e}_t, \mathbf{Z}^{t-1}$ can be found from that of $\begin{pmatrix} \mathbf{X}_t \\ \mathbf{e}_t \end{pmatrix} | \mathbf{Z}^{t-1}$ by conditioning on \mathbf{e}_t (see part (c,i) of Theorem A.4.2), and the distribution of $\begin{pmatrix} \mathbf{X}_t \\ \mathbf{e}_t \end{pmatrix} | \mathbf{Z}^{t-1}$ can be found from those of $\mathbf{X}_t | \mathbf{Z}^{t-1}$ and $\mathbf{e}_t | \mathbf{X}_t, \mathbf{Z}^{t-1}$ by part (c,ii) of Theorem A.4.2. When $t = 1$ we

want the distribution of $\mathbf{X}_1|\mathbf{Z}_1$, so we argue in the same way except there is no \mathbf{Z}^{t-1} .

Since $\mathbf{X}_t = \mathbf{A}_t\mathbf{X}_{t-1} + \mathbf{w}_t$, we have

$$\begin{aligned} \mathbf{E}(\mathbf{X}_t|\mathbf{Z}^{t-1}) &= \mathbf{A}_t\mathbf{E}(\mathbf{X}_{t-1}|\mathbf{Z}^{t-1}) + \mathbf{E}(\mathbf{w}_t|\mathbf{Z}^{t-1}) \\ &= \mathbf{A}_t\hat{\mathbf{X}}_{t-1} \\ \text{Var}(\mathbf{X}_t|\mathbf{Z}^{t-1}) &= \mathbf{A}_t\text{Var}(\mathbf{X}_{t-1}|\mathbf{Z}^{t-1})\mathbf{A}_t^T + \text{Var}(\mathbf{w}_t|\mathbf{Z}^{t-1}) \\ &= \mathbf{A}_t\hat{\Sigma}_{t-1}\mathbf{A}_t^T + \mathbf{W}_t, \end{aligned}$$

that is,

$$\mathbf{X}_t|\mathbf{Z}^{t-1} \sim N_r(\mathbf{A}_t\hat{\mathbf{X}}_{t-1}, \mathbf{A}_t\hat{\Sigma}_{t-1}\mathbf{A}_t^T + \mathbf{W}_t).$$

Since $\mathbf{e}_t = \mathbf{Z}_t - \mathbf{H}_t\mathbf{A}_t\hat{\mathbf{X}}_{t-1}$ and $\mathbf{Z}_t = \mathbf{H}_t\mathbf{X}_t + \mathbf{V}_t$, we have $\mathbf{e}_t = \mathbf{H}_t[\mathbf{X}_t - \mathbf{A}_t\hat{\mathbf{X}}_{t-1}] + \mathbf{v}_t$, and since $\hat{\mathbf{X}}_{t-1}$ is a function of \mathbf{Z}^{t-1} , we have

$$\mathbf{e}_t|\mathbf{X}_t, \mathbf{Z}^{t-1} \sim N_d(\mathbf{H}_t(\mathbf{X}_t - \mathbf{A}_t\hat{\mathbf{X}}_{t-1}), \mathbf{V}_t).$$

Let $\mathbf{R}_t = \mathbf{A}_t\hat{\Sigma}_{t-1}\mathbf{A}_t^T + \mathbf{W}_t$, and to use part (c,ii) of Theorem A.4.2, make the identifications

$$\begin{aligned} \mathbf{X} &\leftrightarrow \mathbf{e}_t, & \mathbf{Y} &\leftrightarrow \mathbf{X}_t, & \mu_Y &\leftrightarrow \mathbf{A}_t\hat{\mathbf{X}}_{t-1}, & \mu_X &\leftrightarrow \mathbf{0}, \\ \Sigma_{XX} &\leftrightarrow \mathbf{H}_t\mathbf{R}_t\mathbf{H}_t^T + \mathbf{V}_t, & \Sigma_{XY} &\leftrightarrow \mathbf{H}_t\mathbf{R}_t, & \Sigma_{YY} &\leftrightarrow \mathbf{R}_t. \end{aligned}$$

To verify that this identification is valid, we suppress reference to \mathbf{Z}^{t-1} and note that

$$\begin{aligned} \mathbf{e}_t|\mathbf{X}_t &\leftrightarrow \mathbf{X}|\mathbf{Y} \\ &\sim N_d(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(\mathbf{Y} - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}) \\ &= N_d(\mathbf{H}_t\mathbf{R}_t\mathbf{R}_t^{-1}(\mathbf{X}_t - \mathbf{A}_t\hat{\mathbf{X}}_{t-1}), \mathbf{H}_t\mathbf{R}_t\mathbf{H}_t^T + \mathbf{V}_t - \mathbf{H}_t\mathbf{R}_t\mathbf{R}_t^{-1}\mathbf{R}_t\mathbf{H}_t^T) \\ &= N_d(\mathbf{H}_t(\mathbf{X}_t - \mathbf{A}_t\hat{\mathbf{X}}_{t-1}), \mathbf{V}_t) \end{aligned}$$

as required. Thus we have by part (c,ii) of Theorem A.4.2,

$$\begin{pmatrix} \mathbf{X}_t \\ \mathbf{e}_t \end{pmatrix} |\mathbf{Z}^{t-1} \sim N_{r+d} \left(\begin{bmatrix} \mathbf{0} \\ \mathbf{A}_t\hat{\mathbf{X}}_{t-1} \end{bmatrix}, \begin{bmatrix} \mathbf{H}_t\mathbf{R}_t\mathbf{H}_t^T + \mathbf{V}_t & \mathbf{H}_t\mathbf{R}_t \\ \mathbf{R}_t\mathbf{H}_t^T & \mathbf{R}_t \end{bmatrix} \right),$$

which in turn gives the desired recursion for $\hat{\mathbf{X}}_t$ and $\hat{\Sigma}_t$ by part (c,i) of Theorem A.4.2.

Part (b) for $h = 0$ follows by definition, while for $h > 0$ the results come from the fact that $\mathbf{X}_{t+h-1} = \mathbf{A}_{t+h-2}\mathbf{X}_{t+h-2} + \mathbf{w}_{t+h-1}$.

Alternate Expression of KFA

There are many algebraically equivalent ways to express the Kalman Filter Algorithm (KFA). In the theorem we wrote it in a way making it clear how the recursion goes from one step to the next, and how it starts. To obtain more intuitively appealing expressions, define

$$\tilde{\mathbf{X}}_t = E(\mathbf{X}_t | \mathbf{Z}^{t-1}) \quad \text{and} \quad \tilde{\Sigma}_t = \text{Var}(\mathbf{X}_t | \mathbf{Z}^{t-1}),$$

that is, the mean and variance of \mathbf{X}_t having only observed \mathbf{Z} up to time $t - 1$. Then clearly,

$$\tilde{\mathbf{X}}_t = \mathbf{A}_t \tilde{\mathbf{X}}_{t-1} \quad \text{and} \quad \tilde{\Sigma}_t = \mathbf{R}_t;$$

that is, our best “estimate” of \mathbf{X}_t without observing \mathbf{Z}_t is $\mathbf{A}_t \tilde{\mathbf{X}}_{t-1}$. If we define the Kalman gain matrix \mathbf{K}_t by

$$\mathbf{K}_t = \tilde{\Sigma}_t \mathbf{H}_t^T (\mathbf{H}_t \tilde{\Sigma}_t \mathbf{H}_t^T + \mathbf{V}_t)^{-1},$$

we can write

$$\hat{\mathbf{X}}_t = \tilde{\mathbf{X}}_t + \mathbf{K}_t (\mathbf{Z}_t - \mathbf{H}_t \tilde{\mathbf{X}}_t);$$

that is, our best guess of \mathbf{X}_t given \mathbf{Z}^t is a linear combination of $\tilde{\mathbf{X}}_t$ (our best guess of \mathbf{X}_t given \mathbf{Z}^{t-1}) and a variable expressing how well $\mathbf{H}_t \tilde{\mathbf{X}}_t$ predicts the new observation \mathbf{Z}_t . Note how similar this is to recursive regression (see Theorem A.5.5). We can summarize the alternative expression for the KFA as $\hat{\mathbf{X}}_0 = E(\mathbf{X}_0)$, $\hat{\Sigma}_0 = \text{Var}(\mathbf{X}_0)$, and

$$\hat{\mathbf{X}}_t = \tilde{\mathbf{X}}_t + \mathbf{K}_t \mathbf{e}_t$$

$$\hat{\Sigma}_t = [\mathbf{I}_r - \mathbf{K}_t \mathbf{H}_t] \tilde{\Sigma}_t,$$

where

$$\tilde{\mathbf{X}}_t = \mathbf{A}_t \tilde{\mathbf{X}}_{t-1}$$

$$\mathbf{e}_t = \mathbf{Z}_t - \mathbf{H}_t \tilde{\mathbf{X}}_t$$

$$\tilde{\Sigma}_t = \mathbf{A}_t \tilde{\Sigma}_{t-1} \mathbf{A}_t^T + \mathbf{W}_t$$

$$\mathbf{K}_t = \tilde{\Sigma}_t \mathbf{H}_t^T (\mathbf{H}_t \tilde{\Sigma}_t \mathbf{H}_t^T + \mathbf{V}_t)^{-1}.$$

A.4.4. Convergence of Random Variables

Many of the theoretical results for statistics calculated in time series analysis are stated as limiting results as the length of the time series gets large. In this section we discuss the various modes of convergence of random variables.

Definition. Let $\{X_n, n = 1, 2, \dots\}$ be a sequence of random variables such that the cdf of X_n is F_n . Let X be another random variable, and suppose that the cdf of X is F . Then

a) X_N converges to the constant c IN PROBABILITY if $\forall \epsilon > 0$,

$$\lim_{n \rightarrow \infty} \Pr(|X_n - c| > \epsilon) = 0.$$

We denote this by $X_n \xrightarrow{P} c$, and say that $X_n \xrightarrow{P} X$ if $X_n - X \xrightarrow{P} 0$.

b) X_n converges to the constant c WITH PROBABILITY ONE (or almost surely) if $\forall \epsilon > 0$,

$$\lim_{N \rightarrow \infty} \Pr\left(\sup_{n \geq N} |X_n - c| > \epsilon\right) = 0.$$

We denote this by $X_n \xrightarrow{a.s.} c$, and say that $X_n \xrightarrow{a.s.} X$ if $X_n - X \xrightarrow{a.s.} 0$.

c) X_n converges to X IN MEAN SQUARE if

$$\lim_{n \rightarrow \infty} E(|X_n - X|^2) = 0.$$

We denote this by $X_n \xrightarrow{m.s.} X$.

d) X_n converges to X IN DISTRIBUTION if

$$\lim_{n \rightarrow \infty} F_n(x) = F(x)$$

for every x where F is continuous. We denote this by $X_n \xrightarrow{L} X$.

e) If $\{Z_n, n = 0, \pm 1, \pm 2, \dots\}$ is a doubly infinite sequence of random variables, and $\{\beta_j, j = 0, \pm 1, \pm 2, \dots\}$ is a sequence of constants, we say that

$$Y_n = \sum_{j=-\infty}^{\infty} \beta_j Z_{n-j}$$

as a limit in mean square if

$$Y_{n,m} = \sum_{j=-M}^M \beta_j Z_{n-j} \xrightarrow{m.s.} Y_n.$$

We note that a statistic $\hat{\theta}_n$, calculated from a realization of length n , is said to be a weakly (strongly) consistent estimator of a parameter θ if $\hat{\theta}_n \xrightarrow{P} \theta$ ($\hat{\theta}_n \xrightarrow{a.s.} \theta$). Also, a random variable X_n is said to be asymptotically normally distributed with asymptotic mean μ_n and asymptotic variance σ_n^2 , if

$$\frac{X_n - \mu_n}{\sigma_n} \xrightarrow{\mathcal{L}} X \sim N(0, 1).$$

We denote such a random variable by $X_n \sim AN(\mu_n, \sigma_n^2)$. For example, the usual Central Limit Theorem of statistics says that $\bar{X} \sim AN(\mu, \sigma^2/n)$.

Convergence of Random Vectors

For a sequence of random vectors $\{\mathbf{X}_n\}$, we can define concepts of convergence analogous to those for a sequence of single random variables. Convergences in probability, with probability one, and in mean square are defined pointwise; that is, we say that $\mathbf{X}_n \xrightarrow{P} \mathbf{X}$, $\mathbf{X}_n \xrightarrow{a.s.} \mathbf{X}$, or $\mathbf{X}_n \xrightarrow{m.s.} \mathbf{X}$ if each element of \mathbf{X}_n converges to the corresponding element of \mathbf{X} . For a random vector to be asymptotically normal we will write $\mathbf{X}_n \sim AN_r(\mu_n, \Sigma_n)$ if Σ_n has nonzero diagonal elements for all n sufficiently large, and if for every vector \mathbf{h} such that $\mathbf{h}^T \Sigma_n \mathbf{h} > 0$ for all n sufficiently large, then

$$\mathbf{h}^T \mathbf{X}_n \sim AN(\mathbf{h}^T \mu_n, \mathbf{h}^T \Sigma_n \mathbf{h}).$$

Thus a random vector is said to be asymptotically normal if all linear functions of it are asymptotically normal in the scalar sense.

Several times in the book (see Problem T3.3 for example), we make use of the fact that a continuous function of an asymptotically normal sequence of random vectors is also asymptotically normal. We state this formally in the following theorem (see Serfling (1980), p. 122, for example).

Theorem A.4.6 CONTINUOUS FUNCTION THEOREM

Suppose that $\mathbf{X}_n \sim AN_r(\mu, b_n \Sigma)$, where Σ is a covariance matrix and $b_n \rightarrow 0$ as $n \rightarrow \infty$. Let

$$\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \dots, g_p(\mathbf{x}))^T, \quad \mathbf{x} = (x_1, \dots, x_p)^T,$$

be a vector-valued function for which each component function $g_i(\mathbf{x})$ is real valued and totally differentiable. Let \mathbf{G} be the $(p \times k)$ matrix whose (i, j) th element is the partial derivative of g_i with respect to x_j , evaluated at the vector μ . Then

$$\mathbf{g}(\mathbf{X}_n) \sim AN(\mathbf{g}(\mu), b_n^2 \mathbf{G} \Sigma \mathbf{G}^T).$$

A.5. Least Squares Regression Analysis

Regression analysis plays two important roles in the study of time series. First, regression is an important tool in the transformation of a time series to remove trends, cycles, or other deterministic functions from data. Just as importantly, much of the theory and methods of time series analysis can be seen as being analogous to the theory and methods of regression analysis. Thus in this section we will try to carefully describe many of the ideas of regression analysis.

A.5.1. The General Linear Model

In a wide variety of statistical settings one is presented with n observations y_1, \dots, y_n on some dependent variable y , and corresponding to y_i there is a vector $\mathbf{x}_i^T = (X_{i1}, \dots, X_{ip})$ of observations on independent variables X_1, \dots, X_p . Usually one seeks to determine models which adequately express the average behavior of y as a linear function of the X 's. We will consider the values of the independent variables to be fixed, that is, nonrandom. If they are in fact random, the material that we will discuss in this section will be considered to be conditional on the observed values of the independent variables.

Let $\mathbf{y} = (y_1, \dots, y_n)^T$ be an n -dimensional random vector satisfying the general linear model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where \mathbf{X} is an $(n \times p)$ matrix of fixed constants, $\boldsymbol{\beta}$ is a $(p \times 1)$ vector of unknown parameters, and $\boldsymbol{\epsilon}$ is an $(n \times 1)$ random vector satisfying $E(\boldsymbol{\epsilon}) = \mathbf{0}_n$, $\text{Var}(\boldsymbol{\epsilon}) = \sigma^2 \mathbf{I}_n$. We will assume that \mathbf{X} is of full rank p , that is, $n \geq p$, and no column of \mathbf{X} is a linear combination of the rest, that is, $\mathbf{X}\mathbf{h} \neq \mathbf{0}_p$ for any $(p \times 1)$ vector \mathbf{h} that is not identically zero. This means that $\mathbf{X}^T\mathbf{X}$ is also of full rank p and is in fact positive definite; that is, if $\mathbf{h} \neq \mathbf{0}_p$, then

$$\mathbf{h}^T \mathbf{X}^T \mathbf{X} \mathbf{h} = (\mathbf{X}\mathbf{h})^T \mathbf{X}\mathbf{h} = \mathbf{z}^T \mathbf{z} > 0,$$

since $\mathbf{z} = \mathbf{X}\mathbf{h} \neq \mathbf{0}_p$.

Thus let $\hat{\boldsymbol{\beta}}$ be the unique vector satisfying the prediction normal equations $\mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\beta}} = \mathbf{X}^T \mathbf{y}$, that is,

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{A} \mathbf{y}.$$

Thus $\hat{\boldsymbol{\beta}}$ is a linear function of the data \mathbf{y} and $\mathbf{A} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ is the matrix of the linear transformation.

Let $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \mathbf{H} \mathbf{y}$ denote the vector of fitted (or predicted) values, and let $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = (\mathbf{I}_n - \mathbf{H}) \mathbf{y}$ denote the vector of residuals. Note that \mathbf{H} is called the hat matrix and that it is idempotent, that is, $\mathbf{H}^2 = \mathbf{H}$ since

$$\mathbf{H}^2 = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T = \mathbf{H}.$$

Also \mathbf{H} and $(\mathbf{I}_n - \mathbf{H})$ are symmetric while $(\mathbf{I}_n - \mathbf{H})$ is idempotent.

Define the residual sum of squares (RSS) by

$$\begin{aligned}
 RSS &= \mathbf{e}^T \mathbf{e} = \mathbf{y}^T (\mathbf{I}_n - \mathbf{H}) (\mathbf{I}_n - \mathbf{H}) \mathbf{y} \\
 &= \mathbf{y}^T (\mathbf{I}_n - \mathbf{H})^2 \mathbf{y} = \mathbf{y}^T (\mathbf{I}_n - \mathbf{H}) \mathbf{y} \\
 &= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{H} \mathbf{y} = \mathbf{y}^T \mathbf{y} - \hat{\mathbf{y}}^T \mathbf{y} \\
 &= \mathbf{y}^T \mathbf{y} - \hat{\boldsymbol{\beta}}^T \mathbf{X}^T \mathbf{y},
 \end{aligned}$$

and let

$$\hat{\sigma}^2 = \frac{RSS}{n - p}.$$

The model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ can be thought of as a decomposition of \mathbf{y} into the sum of two parts, the first being deterministic and the second being random. The deterministic part is something that can be explained or controlled scientifically while the random part is unexplainable and uncontrollable. It is important scientifically to get an idea of how much of the variability in \mathbf{y} is due to each of the two parts of the model. If \mathbf{X} contains a column of ones, that is, the model has an intercept term, then one usually measures the proportion of the variability due to the deterministic part by the multiple correlation coefficient

$$R^2 = 1 - \frac{s_e^2}{s_y^2},$$

where s_e^2 and s_y^2 are the sample variances of \mathbf{e} and \mathbf{y} respectively.

To measure the relative importance of a subset of the independent variables in explaining the variability in \mathbf{y} , one traditionally finds RSS for the model with and without the columns of \mathbf{X} corresponding to the variables in question and then judges whether including the "extra" columns leads to a "significant" reduction in RSS . We can phrase this also in terms of the coefficients of the independent variables. The variability in \mathbf{y} "explained" by $\boldsymbol{\beta}_2$ above and beyond that "explained" by $\boldsymbol{\beta}_1$ is measured by

$$R(\boldsymbol{\beta}_2 | \boldsymbol{\beta}_1) = RSS(\boldsymbol{\beta}_1, \boldsymbol{\beta}_2) - RSS(\boldsymbol{\beta}_1),$$

where $\boldsymbol{\beta}_2$ denotes the subset of $\boldsymbol{\beta}$ being examined, $\boldsymbol{\beta}_1$ represents the other elements of $\boldsymbol{\beta}$, and the notation $RSS(\boldsymbol{\theta})$ denotes the residual sum of squares for the model containing only the independent variables corresponding to the elements of a vector $\boldsymbol{\theta}$.

A.5.2. Results for the General Linear Model

We next summarize the basic properties of the quantities defined above without making any further assumptions about ϵ .

Theorem A.5.1	PROPERTIES OF LEAST SQUARES ESTIMATORS
----------------------	---

Under the conditions described above, we have

a) $\hat{\beta}$ minimizes

$$S(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta);$$

that is, $\hat{\beta}$ is that value of β that minimizes the sum of squares of vertical distances between the elements of \mathbf{y} and $\mathbf{X}\beta$.

$$\text{b) } E(\hat{\beta}) = \beta, \quad \text{Var}(\hat{\beta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}.$$

c) In the class of all unbiased estimators of β that are linear functions of \mathbf{y} , $\hat{\beta}$ is best in terms of variance; that is, if $\tilde{\beta}$ is any other linear function of \mathbf{y} having expectation β , then $\text{Var}(\tilde{\beta}) - \text{Var}(\hat{\beta})$ is positive semidefinite, that is,

$$\mathbf{h}^T[\text{Var}(\tilde{\beta}) - \text{Var}(\hat{\beta})]\mathbf{h} \geq 0$$

for any $\mathbf{h} \neq \mathbf{0}_p$.

d) The fitted value \hat{y}_i and residual e_i are uncorrelated, that is, $\text{Cov}(\hat{y}_i, e_i) = 0$. Thus we can write an orthogonal decomposition of \mathbf{y} by

$$\mathbf{y} = \hat{\mathbf{y}} + \mathbf{e}, \quad \text{Cov}(\hat{\mathbf{y}}, \mathbf{e}) = \mathbf{0},$$

and

$$\text{Var}(\hat{\mathbf{y}}) = \text{Var}(\mathbf{H}\mathbf{y}) = \sigma^2\mathbf{H}^2 = \sigma^2\mathbf{H}$$

$$\text{Var}(\mathbf{e}) = \text{Var}([\mathbf{I}_n - \mathbf{H}]\mathbf{y}) = \sigma^2(\mathbf{I}_n - \mathbf{H})^2 = \sigma^2(\mathbf{I}_n - \mathbf{H})$$

$$\text{Var}(\mathbf{y}) = \text{Var}(\hat{\mathbf{y}}) + \text{Var}(\mathbf{e}) = \sigma^2\mathbf{H} + \sigma^2[\mathbf{I}_n - \mathbf{H}] = \sigma^2\mathbf{I}_n.$$

e) $\hat{\sigma}^2$ is an unbiased estimator of σ^2 .

f) If the random variable z and the vector \mathbf{x} are also related by $E(z) = \mathbf{x}^T\beta$, then the linear function of the observed data \mathbf{y} that has mean $\mathbf{x}^T\beta$ and has the smallest variance in the class of all such estimators is given by

$$\hat{z} = \mathbf{x}^T\hat{\beta},$$

which has variance

$$\sigma_{\mathbf{x}}^2 = \sigma^2 \mathbf{x}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x}.$$

g) The value $\hat{\boldsymbol{\beta}}_{\mathbf{H}, \boldsymbol{\Gamma}}$ of $\boldsymbol{\beta}$ that minimizes $S(\boldsymbol{\beta})$ subject to the k ($\leq p$) linear constraints $\mathbf{H}\boldsymbol{\beta} = \boldsymbol{\Gamma}$ is

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{\mathbf{H}, \boldsymbol{\Gamma}} &= \hat{\boldsymbol{\beta}} + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{H} [\mathbf{H} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{H}^T]^{-1} (\boldsymbol{\Gamma} - \mathbf{H} \hat{\boldsymbol{\beta}}) \\ &= \hat{\boldsymbol{\beta}} + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{H} \mathbf{D}_{\mathbf{H}, \boldsymbol{\Gamma}} \mathbf{e}_{\mathbf{H}, \boldsymbol{\Gamma}}, \end{aligned}$$

while

$$RSS(\mathbf{H}, \boldsymbol{\Gamma}) = S(\hat{\boldsymbol{\beta}}_{\mathbf{H}, \boldsymbol{\Gamma}}) = RSS + \mathbf{e}_{\mathbf{H}, \boldsymbol{\Gamma}}^T \mathbf{D}_{\mathbf{H}, \boldsymbol{\Gamma}} \mathbf{e}_{\mathbf{H}, \boldsymbol{\Gamma}},$$

where

$$\mathbf{e}_{\mathbf{H}, \boldsymbol{\Gamma}} = \boldsymbol{\Gamma} - \mathbf{H} \hat{\boldsymbol{\beta}} \quad \text{and} \quad \mathbf{D}_{\mathbf{H}, \boldsymbol{\Gamma}} = [\mathbf{H} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{H}^T]^{-1}.$$

Implications: Part (a) explains why $\hat{\boldsymbol{\beta}}$ is called the least squares estimator, while parts (b) and (c) explain statistically why we use $\hat{\boldsymbol{\beta}}$, namely that it is BLUE (best linear unbiased estimator). Part (d) gives further reason for using $\hat{\boldsymbol{\beta}}$, namely that it affords an orthogonal decomposition of the variability in \mathbf{y} into that due to the deterministic and random parts. Part (e) provides a means of estimating σ^2 unbiasedly, while part (f) shows how to “predict” an unobserved value of the dependent variable when given a new set of values for the independent variables. Finally, part (g) gives expressions for the so-called restricted least squares estimator and its variance.

The next two theorems allow us to perform tests of hypotheses or form confidence intervals if either the errors $\boldsymbol{\epsilon}$ are normally distributed (Theorem A.5.2) or certain conditions on \mathbf{y} and \mathbf{X} are satisfied that provide a central limit type theorem for regression (Theorem A.5.3). In Theorem A.5.2, we will phrase most of the results in terms of a linear function $\boldsymbol{\theta} = \mathbf{A}\boldsymbol{\beta}$ of $\boldsymbol{\beta}$. This allows for more general results, but all of these results can be expressed about $\boldsymbol{\beta}$ if $\mathbf{A} = \mathbf{I}_p$.

Theorem A.5.2 INFERENCES FOR NORMAL ERRORS

Let \mathbf{A} be a full rank ($k \times p$) matrix of constants, and let $\boldsymbol{\theta} = \mathbf{A}\boldsymbol{\beta}$, and $\hat{\boldsymbol{\theta}} = \mathbf{A}\hat{\boldsymbol{\beta}}$. If $\boldsymbol{\epsilon} \sim N_n(0_n, \sigma^2 \mathbf{I}_n)$, then

a) $\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimator of $\boldsymbol{\theta}$,

$$\hat{\boldsymbol{\theta}} \sim N_k(\boldsymbol{\theta}, \sigma^2 \mathbf{A} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{A}^T),$$

and $\hat{\boldsymbol{\theta}}$ is MVUE; that is, of all functions (not just linear functions) of \mathbf{y} that are unbiased for $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}}$ has minimum variance.

b) If we define the quadratic form

$$C_{\bullet}(\hat{\boldsymbol{\theta}}) = (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T \mathbf{V}^{-1} (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}),$$

where $\mathbf{V} = \mathbf{A}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{A}^T = \frac{1}{\sigma^2} \text{Var}(\hat{\boldsymbol{\theta}})$, then we have

$$C_1 = \frac{(n-p)\hat{\sigma}^2}{\sigma^2} \sim \chi_{n-p}^2$$

$$C_2 = \frac{C_{\bullet}(\hat{\boldsymbol{\theta}})}{\sigma^2} \sim \chi_k^2,$$

and C_1 and C_2 are independent. This gives

$$F = \frac{C_2/k}{C_1/(n-p)} = \frac{C_{\bullet}(\hat{\boldsymbol{\theta}})}{k\hat{\sigma}^2} \sim F_{k, n-p},$$

which in turn gives by Scheffé's theorem (see Theorem A.4.3) that the probability is $1 - \alpha$ that simultaneously for all k -dimensional vectors \mathbf{h} ,

$$\mathbf{h}^T \hat{\boldsymbol{\theta}} - S_1(\mathbf{h}) \leq \mathbf{h}^T \boldsymbol{\theta} \leq \mathbf{h}^T \hat{\boldsymbol{\theta}} + S_1(\mathbf{h})$$

or

$$\mathbf{h}^T \hat{\boldsymbol{\theta}} - S_2(\mathbf{h}) \leq \mathbf{h}^T \boldsymbol{\theta} \leq \mathbf{h}^T \hat{\boldsymbol{\theta}} + S_2(\mathbf{h}),$$

where

$$S_1^2(\mathbf{h}) = \frac{\sigma^2 \chi_{\alpha, k}^2}{\mathbf{h}^T \mathbf{V}^{-1} \mathbf{h}}$$

$$S_2^2(\mathbf{h}) = \frac{k \hat{\sigma}^2 F_{\alpha, k, n-p}}{\mathbf{h}^T \mathbf{V}^{-1} \mathbf{h}}.$$

c) If the matrix \mathbf{A} is such that $\mathbf{A}\boldsymbol{\beta} = \boldsymbol{\beta}_2$ where $\boldsymbol{\beta}_2$ consists of k ($< p$) of the elements of $\boldsymbol{\beta}$, and we let $\boldsymbol{\beta}_1$ represent the other $p - k$ elements of $\boldsymbol{\beta}$, then

$$\begin{aligned} F &= \frac{[RSS(\boldsymbol{\beta}_1) - RSS(\boldsymbol{\beta})]/k}{RSS(\boldsymbol{\beta})/(n-p)} \\ &= \frac{R(\boldsymbol{\beta}_2|\boldsymbol{\beta}_1)/k}{\hat{\sigma}^2}, \end{aligned}$$

while if β_2 consists of a single element of β , β_r say, then $F = t^2$, where if we denote the r th diagonal element of $(\mathbf{X}^T \mathbf{X})^{-1}$ by $(\mathbf{X}^T \mathbf{X})_{rr}^{-1}$, we have

$$t = \frac{\hat{\beta}_r - \beta_r}{\sqrt{\hat{\sigma}^2 (\mathbf{X}^T \mathbf{X})_{rr}^{-1}}} \sim t_{n-p}.$$

d) If $\mathbf{H}\beta = \Gamma$, then

$$F = \frac{[RSS(\mathbf{H}, \Gamma) - RSS(\beta)]/k}{RSS(\beta)/(n-p)} \sim F_{k, n-p}.$$

e) If \hat{z} is as in part (f) of Theorem A.5.1, then

$$\hat{z} \sim N(\mathbf{x}^T \beta, \sigma^2 \mathbf{x}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x})$$

$$\hat{z} - z \sim N(0, \sigma^2 (1 + \mathbf{x}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x})).$$

Implications: Part (a) says that the least squares estimator $\hat{\theta}$ is the best unbiased estimator of θ . Parts (b), (c), and (d) allow us to perform tests of hypotheses and form confidence regions for regression parameters. For example, to test $H_0 : \mathbf{A}\beta = \theta_0$ we would reject H_0 if $F > F_{\alpha, k, n-p}$, while a confidence region for θ is given by the set of vectors \mathbf{a} satisfying $C_{\mathbf{a}}(\hat{\theta})/k\hat{\sigma}^2 < F_{\alpha, k, n-p}$. Results similar to the simultaneous confidence intervals in part (b) are used in Chapter 3 to find simultaneous confidence bands for autoregressive spectra (see Theorem 3.8.4). These Scheffé intervals allow the data to suggest interesting functions of the parameters to test. Part (e) provides confidence intervals for the mean $E(z) = \mathbf{x}^T \beta$ of an unobserved value z of the dependent variable corresponding to a new value \mathbf{x} of the independent variables:

$$\hat{z} \pm t_{\alpha/2, n-p} \sqrt{\hat{\sigma}^2 \mathbf{x}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x}}$$

and prediction intervals for z :

$$\hat{z} \pm t_{\alpha/2, n-p} \sqrt{\hat{\sigma}^2 (1 + \mathbf{x}^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x})}.$$

The Simple Form of the REG Command

Most of the material in this section can be illustrated by using the general form of the REG command which we will introduce in Section A.5.5. For the linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ where \mathbf{X} is $(n \times p)$, if the arrays \mathbf{y} and \mathbf{X} contain \mathbf{y} and \mathbf{X} respectively, then the command

REG($\mathbf{y}, \mathbf{X}, n, p, \text{beta}, \text{RSS}, \mathbf{t}, \mathbf{e}$)

will return $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$, $\text{RSS} = \mathbf{y}^T \mathbf{y} - \hat{\boldsymbol{\beta}}^T \mathbf{X}^T \mathbf{y}$, $\mathbf{e} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$, and

$$t_j = \frac{\hat{\beta}_j}{\sqrt{\frac{\text{RSS}}{n-p} (\mathbf{X}^T \mathbf{X})_{jj}^{-1}}}, \quad j = 1, \dots, p$$

in beta , RSS , \mathbf{e} , and \mathbf{t} respectively. See Section A.5.4 for an example of the use of this command.

The Case of Nonnormal Errors

If the errors in the general linear model are not normally distributed, we would hope that $\hat{\boldsymbol{\beta}}$ might still have a normal distribution for large n . After all, $\hat{\beta}_j$ is a linear function of y_1, \dots, y_n and thus of $\epsilon_1, \dots, \epsilon_n$, and we know from elementary statistics that some linear functions of nonnormal random variables are approximately normally distributed. For example, the Central Limit Theorem is concerned with linear functions whose coefficients are all $1/n$.

To obtain a Central Limit Theorem for regression analysis, we must make assumptions both about $\boldsymbol{\epsilon}$ and about the linear function used to calculate $\hat{\boldsymbol{\beta}}$, that is, about $\mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$. There are several sets of conditions in the literature, but we will follow Anderson (1971), p. 23.

Theorem A.5.3	CENTRAL LIMIT THEOREM FOR REGRESSION
----------------------	---

Let $y_t = \mathbf{x}_t^T \boldsymbol{\beta} + \epsilon_t$, for $t = 1, 2, \dots$, where the ϵ 's are iid with mean zero and variance σ^2 , and $\mathbf{x}_1, \mathbf{x}_2, \dots$ are $(p \times 1)$ vectors of fixed constants. For $n = 1, 2, \dots$, let \mathbf{X}_n be the $(n \times p)$ matrix having t th row \mathbf{x}_t^T , and let a_{ni} be the i th diagonal element of $\mathbf{X}_n^T \mathbf{X}_n$, for $i = 1, \dots, p$. Let

$$\mathbf{D}_n = \mathbf{D}_n^{-1} \mathbf{X}_n^T \mathbf{X}_n \mathbf{D}_n^{-1},$$

where

$$\mathbf{D}_n = \text{Diag}(a_{n1}^{1/2}, \dots, a_{np}^{1/2});$$

that is, the (i, j) th element of \mathbf{R}_n is

$$R_{n,ij} = \frac{(\mathbf{X}_n^T \mathbf{X}_n)_{ij}}{\sqrt{(\mathbf{X}_n^T \mathbf{X}_n)_{ii}(\mathbf{X}_n^T \mathbf{X}_n)_{jj}}},$$

that is, \mathbf{R}_n is just a scaled form of $\mathbf{X}_n^T \mathbf{X}_n$.

Let $\hat{\beta}_n$ and $\hat{\sigma}_n^2$ be the ordinary least squares estimators of β and σ^2 based on \mathbf{X}_n and $\mathbf{y}_n = (y_1, \dots, y_n)^T$. If

$$i) \lim_{n \rightarrow \infty} a_{ni} = \infty, \quad i = 1, \dots, p,$$

$$ii) \lim_{n \rightarrow \infty} \frac{x_{n+1,i}}{a_{ni}} = 0, \quad i = 1, \dots, p,$$

and there exists a $(p \times p)$ nonsingular matrix \mathbf{R} such that

$$iii) \lim_{n \rightarrow \infty} R_{n,ij} = R_{ij}, \quad i, j = 1, \dots, p,$$

then

$$a) \mathbf{D}_n(\hat{\beta}_n - \beta) \xrightarrow{\mathcal{L}} N_p(0_p, \sigma^2 \mathbf{R}^{-1}).$$

$$b) \hat{\sigma}_n^2 \xrightarrow{P} \sigma^2 \text{ and } \hat{\sigma}_n^2 \text{ is asymptotically independent of } \hat{\beta}_n.$$

Implications: These results essentially allow us to use all of the tests and confidence regions in Theorem A.5.2 except that now significance and confidence levels are valid only for large n . The conditions (i) and (ii) ensure that the weights applied to the ϵ 's do not die out too quickly as $n \rightarrow \infty$. The matrix \mathbf{D}_n plays the role that \sqrt{n} does in the usual Central Limit Theorem; that is, $\sqrt{n}(\bar{X} - \mu) \xrightarrow{\mathcal{L}} N(0, \sigma^2)$.

A.5.3. The Case of Correlated Errors

If $\mathbf{y} = \mathbf{X}\beta + \epsilon$, where $\text{Var}(\epsilon) = \sigma^2 \mathbf{V}$ and \mathbf{V} is a known $(n \times n)$ positive definite matrix, the results of the previous two theorems can still be applied by transforming the original model to one having uncorrelated errors.

If we know \mathbf{V} , we can calculate its modified Cholesky decomposition (see Section A.1.1) $\mathbf{V} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ and its inverse square root $\mathbf{V}^{-1/2}$. If we define $\mathbf{z} = \mathbf{V}^{-1/2}\mathbf{y}$, $\mathbf{W} = \mathbf{V}^{-1/2}\mathbf{X}$, and $\eta = \mathbf{V}^{-1/2}\epsilon$, then we have

$$\mathbf{z} = \mathbf{W}\beta + \eta,$$

where $\text{Var}(\eta) = \sigma^2 \mathbf{I}_n$, and all of the results of the previous theorems can be applied to this new model. We summarize some of these results in the next theorem.

Theorem A.5.4

GENERALIZED LEAST SQUARES

Let $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim (0_n, \sigma^2 \mathbf{V})$. Then

a) $\tilde{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{V}^{-1} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{V}^{-1} \mathbf{y}$ is BLUE for $\boldsymbol{\beta}$,

$$\text{Var}(\tilde{\boldsymbol{\beta}}) = \sigma^2 (\mathbf{X}^T \mathbf{V}^{-1} \mathbf{X})^{-1},$$

and

$$\tilde{\sigma}^2 = \frac{\mathbf{y}^T \mathbf{V}^{-1} \mathbf{y} - \tilde{\boldsymbol{\beta}}^T \mathbf{X}^T \mathbf{y}}{n - p}$$

is an unbiased estimator of σ^2 .

b) If $\boldsymbol{\epsilon} \sim N_n(0_n, \sigma^2 \mathbf{V})$, then

$$\tilde{\boldsymbol{\beta}} \sim N_p(\boldsymbol{\beta}, \sigma^2 (\mathbf{X}^T \mathbf{V}^{-1} \mathbf{X})^{-1})$$

$$C_{\boldsymbol{\beta}}(\tilde{\boldsymbol{\beta}}) = \frac{1}{\sigma^2} (\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta})^T \mathbf{X}^T \mathbf{V}^{-1} \mathbf{X} (\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}) \sim \chi_p^2,$$

and $C_{\boldsymbol{\beta}}(\tilde{\boldsymbol{\beta}})$ is independent of

$$\frac{(n - p) \tilde{\sigma}^2}{\sigma^2} \sim \chi_{n-p}^2.$$

If we don't know \mathbf{V} but we do know that its elements are a continuous function of a finite number r of parameters $\boldsymbol{\theta}$, that is, $\mathbf{V} = \mathbf{V}(\boldsymbol{\theta})$, and if consistent estimators $\hat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$ are available, then we have for example

$$C_{\boldsymbol{\beta}}(\tilde{\boldsymbol{\beta}}) = \frac{1}{\sigma^2} (\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta})^T \mathbf{X}^T \mathbf{V}^{-1}(\hat{\boldsymbol{\theta}}) \mathbf{X} (\tilde{\boldsymbol{\beta}} - \boldsymbol{\beta}) \rightarrow \chi_p^2.$$

This idea is used in Section 3.7.1 to analyze regression models having autoregressive errors.

A.5.4. Special Types of Regression Analyses

In this subsection we consider orthogonal, stepwise, and recursive regression. Each of these is useful in analyzing ordinary regression data but is also important because each has an analog in time series analysis.

Orthogonal Regression

A source of difficulty in regression analysis is that the elements of $\hat{\beta}$ are correlated since $\text{Var}(\hat{\beta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$ which is not generally diagonal, and thus inferences made about one element are not independent of inferences made about other elements. One strategy for solving this problem is given by orthogonalizing \mathbf{X} . The Gram-Schmidt decomposition $\mathbf{X} = \mathbf{Q}\mathbf{R}$ (Section A.1.2) can be used to do this because if we write $\boldsymbol{\theta} = \mathbf{R}\boldsymbol{\beta}$, then we have $\mathbf{y} = \mathbf{Q}\boldsymbol{\theta} + \boldsymbol{\epsilon}$ and $\hat{\boldsymbol{\beta}}$ satisfies

$$\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{y}$$

$$\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R}\hat{\boldsymbol{\beta}} = \mathbf{R}^T\mathbf{Q}^T\mathbf{y}$$

$$\mathbf{D}\mathbf{R}\hat{\boldsymbol{\beta}} = \mathbf{Q}^T\mathbf{y},$$

where $\mathbf{D} = \mathbf{Q}^T\mathbf{Q}$. Further,

$$\hat{\boldsymbol{\theta}} = (\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\mathbf{y} = \mathbf{D}^{-1}\mathbf{Q}^T\mathbf{y} = \mathbf{R}\hat{\boldsymbol{\beta}}$$

and

$$\text{Var}(\hat{\boldsymbol{\theta}}) = \sigma^2(\mathbf{Q}^T\mathbf{Q})^{-1} = \sigma^2\mathbf{D}^{-1},$$

which is diagonal, and thus the elements of $\hat{\boldsymbol{\theta}}$ are uncorrelated (independent if the errors are normal). Thus one can make inferences separately about its elements and then recover $\hat{\boldsymbol{\beta}} = \mathbf{R}^{-1}\hat{\boldsymbol{\theta}}$. In Section 3.4.4 we see that sample partial autocorrelations in time series analysis consist of the orthogonal parameters in a special kind of regression problem.

Stepwise Regression

Perhaps the most important use of the sweep operator (see Section A.1.3) is in stepwise regression. Given n observations on a dependent variable y and independent variables X_1, \dots, X_p , the aim of stepwise regression is to determine a subset of the independent variables that adequately explains the variability in y . This is usually done by (see Jennrich (1977) for example) recursively adding and deleting independent variables from the model.

The procedure begins with no variables in the model. At each step of the algorithm, a variable that is a contender to be added and one to be deleted is chosen (at the first step there is no deletion contender). Then the algorithm judges whether the deletion contender should in fact be deleted. If so, it is deleted and the algorithm goes to the next step. If not, then the algorithm checks whether the addition contender should be added. If so, it is added and the algorithm goes to the next step. This stepping stops when no more variables can be added or deleted.

The sweep algorithm is ideal for carrying out the steps described above. The algorithm begins by forming the matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{X}^T \mathbf{X} & \mathbf{X}^T \mathbf{y} \\ \mathbf{y}^T \mathbf{X} & \mathbf{y}^T \mathbf{y} \end{bmatrix}.$$

The adding and deleting of variables are done recursively in \mathbf{A} . Adding a variable that is not in the model is done by sweeping \mathbf{A} on the diagonal corresponding to its column in the \mathbf{X} matrix. Deleting a variable that is in the model is done by sweeping on its diagonal element.

Denote the matrix \mathbf{A} after k steps of the algorithm by

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{u} \\ \mathbf{v}^T & d \end{bmatrix}.$$

Then d is the residual sum of squares for the model chosen so far, while \mathbf{u} and \mathbf{v} are equal except that the elements corresponding to variables that are in the model have opposite signs. The elements of \mathbf{v} have the correct sign. Thus at the k th step, suppose variables having indices k_1, \dots, k_r are in the model while those having k_{r+1}, \dots, k_p are not. Let k_D and k_A denote the contenders for deletion and addition respectively. Then k_A and k_D are the values of j that maximize

$$\frac{|u_{k_j}|}{B_{k_j, k_j}}, \quad 1 \leq j \leq r,$$

and minimize

$$\frac{|u_{k_j}|}{B_{k_j, k_j}}, \quad r+1 \leq j \leq p,$$

respectively. There are a variety of rules for addition and deletion but the one used in TIMESLAB is given by adding k_A if

$$C_A = \frac{\chi_{\alpha, p-r}^2 \sum_{j=r+1}^p u_{k_j}^2}{RSS/(n-r-1)} > 1$$

and deleting k_D if

$$C_D = \frac{\chi_{.01, 1}^2 |u_{k_D}|}{RSS/(n-r-1)} < 1,$$

where $\chi_{\alpha, p-r}^2$ is the value of a χ_{p-r}^2 random variable having area α to its right.

Most implementations of stepwise regression also will not allow deletion of the most recently added variable, and will have a maximum number of steps to make sure that no infinite loop of adding and deleting can occur.

Stepwise regression is very popular, but proving any sort of theoretical results such as p -values or confidence limits is very difficult. Traditionally it is regarded as an exploratory technique whose results must be confirmed statistically by obtaining more data.

The Stepwise Form of the REG Command

If the arrays **y** and **X** contain the observations and design matrix for an $(n \times p)$ regression model, then the command

REG(y,X,n,p,pval,kopt,k,beta,ind,RSS)

will perform the stepwise regression procedure described above. The first six arguments are input. The argument **pval** corresponds to $1-\alpha$ in the description above, while **kopt** is zero if the usual algorithm is desired, or a positive integer **j** if the user wants to “force into” the model the variables having indices contained in the array **ind** of length **j**. In either case, the output integer variable **k** is the number of variables in the chosen model (**k=j** if **kopt>0**), while **ind** and **beta** are output arrays of length **k** having the indices and coefficients in the chosen model. Finally, **RSS** is the residual sum of squares for the chosen model.

An Example of the Simple and Stepwise Forms of REG

To illustrate how the simple and stepwise forms of the **REG** commands can be used, consider the Hald data (Draper and Smith (1981), p. 296) given in Table A.1. Note that this table shows the data set exactly as it appears in the disk file called **HALD.DAT**. The first four columns are independent variables while the last is the dependent variable.

Table A.1. The Hald Data

Hald Data (X1,X2,X3,X4,Y)				
65				
7	26	6	60	78.5
1	29	15	52	74.3
11	56	8	20	104.3
11	31	8	47	87.6
7	52	6	33	95.9
11	55	9	22	109.2
3	71	17	6	102.7
1	31	22	44	72.5
2	54	18	22	93.1
21	47	4	26	115.9
1	40	23	34	83.9
11	66	9	12	113.3
10	68	8	12	109.4

The following macro fits both the full model using the simple form and a subset model using the stepwise form.

```

1 ;;
2 ;;   REG.MAC: macro to do regression analysis of data on the file
3 ;;   fname. In that file, you need to put one observation
4 ;;   per line, with the X's first and then the y as the last
5 ;;   entry in the line. The first line of the file should
6 ;;   be a title and the second line should be n*(m+1) where
7 ;;   n is the number of observations and m is the number
8 ;;   of regressors (not including an intercept). Note that
9 ;;   there should be no column of 1's in the file.
10 ;;
11 ;;   INPUT: fname, n, m, pval, kopt
12 ;;
13 PAUSE
14 ;start
15 PROMPTOFF
16 READ(fname,X,nn)      ;nn should be n*(m+1)
17 mp1=m+1                ;
18 X=TRANS(X,mp1,n)      ;this will get x in the right form
19 one=LINE(n,1,0)       ;put intercept in
20 X=<one,X>
21 nnpi=nn+1
22 nnpn=nn+n
23 y=EXTRACT(X,nnpi,nnpn) ;pull off y from end of X
24 REG(y,X,n,mp1,beta,rss,t,e) ;simple form
25 LIST(beta)            ;list coefficients
26 LIST(t)               ;list t-statistics
27 LIST(e,n,7,7f8.3)     ;list residuals, 7 per line
28 ;
29 ;form and list estimate of covariance matrix:
30 ;
31 XTX=MMULT(X,X,n,mp1)
32 VAR=MINV(XTX,mp1,ier)
33 df=n-mp1
34 s2=rss/df
35 VAR=s2*VAR
36 label(VAR)='Covariance Matrix of Estimates'
37 mp12=mp1*mp1
38 LIST(VAR,mp12,mp1,5f10.4)
39 PAUSE
40 ;
41 ;Stepwise form:
42 ;
43 REG(y,X,n,mp1,pval,kopt,k,betaa,ind,rssa)
44 IF(k.eq.0,end)         ;if no variables chosen,
45 LIST(ind,k)            ;end macro. Otherwise, list
46 LIST(betaa,k)          ;output.
47 XA=EXTRACT(X,n,1,1,ind,k) ;Pull off columns of chosen
48 XTXA=MMULT(XA,XA,n,k)   ;variables.
49 VARA=MINV(XTXA,k,ier)
50 dfa=n-k
51 s2a=rssa/dfa
52 VARA=s2a*VARA
53 label(VAR)='Covariance Matrix of Estimates'
54 k2=k*k

```

```
55 LIST(VARA,k2,k,5f10.4)
56 ;
57 ;end
58 ;
59 PROMPTON
```

If we precede the invoking of the macro with the lines

```
fname='hald'
n=13
m=4
kopt=0
pval=.95
```

then the output generated by the stepwise form of the REG command is given in Table A.2.

Note that the macro inserts a column of ones into the regression matrix and thus there is a total of five independent variables. These are labeled 1 through 5 in the stepwise output. The stepwise regression begins with three lines that tell how many variables are in the full model and the maximum number of variables that can be in the final model. This maximum number is actually displayed as 0 if *kopt* is 0. Each step of the stepwise procedure is displayed between a pair of dashed lines. Included are the number of the variables and the value of the corresponding coefficients that are already in the model (there are no such variables during the first step), the value of *RSS* prior to beginning the step (this is labeled *RVAR*), and seven other variables. The variables labeled *V ADD* and *V DEL* are the numbers of the variables that are contenders for addition and deletion in this step. The variables labeled *NPRED* and *NO CYC* are the number of variables in the model prior to this step and the number of steps that have been made prior to this step respectively. Thus at the first step these are both 0. Finally, the variables labeled *C ADD* and *C DEL* are the criteria for addition and deletion; that is, they are C_A and C_D in our discussion of stepwise regression above. At a given step, the variable labeled *V ADD* can be added if the criterion is greater than 1, while the variable labeled *V DEL* can be deleted if the criterion is less than 1. Recall that the contender for deletion is checked first. Thus in the next to last step in Table A.2, the variable number 5 is in fact deleted. The stepwise procedure stopped in this example because it could not delete variable 2 because the criterion is 19.611 (which is greater than 1), and it could not add variable 5 because it had just deleted it. Note that the model determined here is the same as that found by Draper and Smith (1981), p. 310.

Table A.2. Stepwise Regression Analysis of the Hald Data

1	REGRESSION ESTIMATION STAGewise SUMMARY							
2	NUMBER OF VARIABLES IN FULL MODEL = 5							
3	MAXIMUM NUMBER OF COEFFICIENTS = 0							
4	-----							
5	VAR IN MODEL			COEFF				
6	C ADD	C DEL	V ADD	V DEL	V LST	MPRED	NO CYC	RVAR
7	4.373	1.493	1	0	0	0	0	0*****
8	-----							
9	VAR IN MODEL			COEFF				
10			1	95.43077				
11	C ADD	C DEL	V ADD	V DEL	V LST	MPRED	NO CYC	RVAR
12	2.504	61.098	5	0	1	1		12713.4470
13	-----							
14	VAR IN MODEL			COEFF				
15			1	117.57210				
16			5	-.73804				
17	C ADD	C DEL	V ADD	V DEL	V LST	MPRED	NO CYC	RVAR
18	2.216	3.129	2	5	5	2	2	882.1382
19	-----							
20	VAR IN MODEL			COEFF				
21			1	103.11730				
22			2	1.43839				
23			5	-.61397				
24	C ADD	C DEL	V ADD	V DEL	V LST	MPRED	NO CYC	RVAR
25	1.002	14.641	3	2	2	3	3	74.7965
26	-----							
27	VAR IN MODEL			COEFF				
28			1	71.89907				
29			2	1.45028				
30			3	.41306				
31			5	-.23933				
32	C ADD	C DEL	V ADD	V DEL	V LST	MPRED	NO CYC	RVAR
33	.006	.253	4	5	3	4	4	48.3989
34	-----							
35	VAR IN MODEL			COEFF				
36			1	52.60338				
37			2	1.46684				
38			3	.66210				
39	C ADD	C DEL	V ADD	V DEL	V LST	MPRED	NO CYC	RVAR
40	.519	19.611	5	2	5	3	5	58.5657
41	-----							
42	Regressors Chosen							
43	1	1.000000		2.000000		3.000000		
44	Stepwise Regression Coefficients							
45	1	52.603380		1.466842		.662097		
46	Covariance Matrix of Estimates							
47	1	4.3567	-.0405	-.0765				
48	4	-.0405	.0123	-.0011				
49	7	-.0765	-.0011	.0018				

Recursive Regression

An important topic in statistics is recursive estimation. In regression analysis this means given the results $\hat{\beta}$, RSS , and $\mathbf{V} = (\mathbf{X}^T \mathbf{X})^{-1}$ of regressing the $(m \times 1)$ vector \mathbf{y} on the $(m \times p)$ matrix \mathbf{X} , we seek the corresponding quantities after adding or deleting data \mathbf{Y}_1 , and \mathbf{X}_1 of length n . Thus we seek

$$\hat{\beta}_N = (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} (\mathbf{X}^T \mathbf{y} \pm \mathbf{X}_1^T \mathbf{y}_1)$$

$$RSS_N = (\mathbf{y}^T \mathbf{y} \pm \mathbf{y}_1^T \mathbf{y}_1) - \hat{\beta}_N^T (\mathbf{X}^T \mathbf{y} \pm \mathbf{X}_1^T \mathbf{y}_1)$$

$$\mathbf{V}_N = (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1},$$

where the \pm is $+$ for adding observations and $-$ for deleting. From the matrix inversion lemma (Theorem A.1.4) we can obtain recursive formulas for these quantities.

Theorem A.5.5	RECURSIVE REGRESSION FORMULAS
----------------------	--------------------------------------

The updated quantities $\hat{\beta}_N$, RSS_N , and \mathbf{V}_N defined above are given by:

$$\begin{aligned}\hat{\beta}_N &= \hat{\beta} \pm \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{e}_1 \\ &= \hat{\beta} \pm (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} \mathbf{X}_1^T \mathbf{e}_1\end{aligned}$$

$$\begin{aligned}RSS_N &= RSS \pm \mathbf{e}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{e}_1 \\ &= RSS \pm \mathbf{e}_1^T [\mathbf{I}_n \mp \mathbf{X}_1 (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} \mathbf{X}_1^T] \mathbf{e}_1\end{aligned}$$

$$\begin{aligned}\mathbf{V}_N &= \mathbf{A} \mp \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{X}_1 \mathbf{A} \\ &= (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1},\end{aligned}$$

where

$$\mathbf{H}_1 = \mathbf{X}_1 \mathbf{A} \mathbf{X}_1^T, \quad \mathbf{A} = (\mathbf{X}^T \mathbf{X})^{-1}, \quad \mathbf{e}_1 = \mathbf{y}_1 - \mathbf{X}_1 \hat{\beta}.$$

Proof: The basis of this theorem is the matrix inversion lemma and the facts (easily verified by multiplication):

$$(\mathbf{I}_n \pm \mathbf{H}_1)^{-1} = \mathbf{I}_n \mp (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{H}_1 = \mathbf{I}_n \mp \mathbf{H}_1 (\mathbf{I}_n \pm \mathbf{H}_1)^{-1}.$$

Thus

$$\begin{aligned}\hat{\beta}_N &= (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} (\mathbf{X}^T \mathbf{y} \pm \mathbf{X}_1^T \mathbf{y}_1) \\ &= [\mathbf{A} \mp \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{X}_1 \mathbf{A}] (\mathbf{X}^T \mathbf{y} \pm \mathbf{X}_1^T \mathbf{y}_1) \\ &= \hat{\beta} \mp \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \hat{\mathbf{y}}_1 \pm \mathbf{A} \mathbf{X}_1^T \mathbf{y}_1 - \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{H}_1 \mathbf{y}_1,\end{aligned}$$

where $\hat{\mathbf{y}}_1 = \mathbf{X}_1 \mathbf{A} \mathbf{X}_1^T \mathbf{y} = \mathbf{X}_1 \hat{\boldsymbol{\beta}}$, which gives

$$\begin{aligned} \hat{\boldsymbol{\beta}}_N - \hat{\boldsymbol{\beta}} &= \pm \mathbf{A} \mathbf{X}_1^T [\mathbf{I}_n \mp (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{H}_1] \mathbf{y}_1 \\ &\mp \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \hat{\mathbf{y}}_1 \\ &= \pm \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} (\mathbf{y}_1 - \hat{\mathbf{y}}_1), \end{aligned}$$

which gives the first expression for $\hat{\boldsymbol{\beta}}_N$. To show the second, note that

$$\begin{aligned} (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} \mathbf{X}_1^T \mathbf{e}_1 &= [\mathbf{A} \mp \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{X}_1 \mathbf{A}] \mathbf{X}_1^T \mathbf{e}_1 \\ &= \mathbf{A} \mathbf{X}_1^T [\mathbf{I}_n \mp (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{H}_1] \mathbf{e}_1 \\ &= \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{e}_1. \end{aligned}$$

To find RSS_N we have that $\pm \mathbf{H}_1 \mathbf{y}_1 = (\mathbf{I}_n \pm \mathbf{H}_1) \mathbf{y}_1 - \mathbf{y}_1$, and so

$$\begin{aligned} RSS_N &= (\mathbf{y}^T \mathbf{y} \pm \mathbf{y}_1^T \mathbf{y}_1) - [\hat{\boldsymbol{\beta}}^T \pm \mathbf{e}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{X}_1 \mathbf{A}] (\mathbf{X}^T \mathbf{y} \pm \mathbf{X}_1^T \mathbf{y}_1) \\ &= RSS \pm \mathbf{y}_1^T \mathbf{y}_1 \mp \hat{\boldsymbol{\beta}}^T \mathbf{X}_1^T \mathbf{y}_1 \mp \mathbf{e}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} (\mathbf{X}_1 \hat{\boldsymbol{\beta}} \pm \mathbf{H}_1 \mathbf{y}_1) \\ &= RSS \pm \mathbf{y}_1^T \mathbf{y}_1 \mp \hat{\boldsymbol{\beta}}^T \mathbf{X}_1^T \mathbf{y}_1 \mp \mathbf{e}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{y}_1^T \\ &\mp \mathbf{e}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} [(\mathbf{I}_n \pm \mathbf{H}_1) \mathbf{y}_1 - \mathbf{y}_1] \\ &= RSS \pm (\mathbf{y}_1^T - \hat{\boldsymbol{\beta}}^T \mathbf{X}_1^T - \mathbf{e}_1^T) \mathbf{y}_1 \mp \mathbf{e}_1^T (\mathbf{I}_n \pm \mathbf{H}_1) (\mathbf{y}_1^T - \mathbf{y}_1) \\ &= RSS \pm \mathbf{e}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{e}_1 \end{aligned}$$

since $\mathbf{e}_1^T = \mathbf{y}_1^T - \hat{\boldsymbol{\beta}}^T \mathbf{X}_1^T$. To obtain the second expression for RSS_N , note that

$$\begin{aligned} \mathbf{e}_1^T [\mathbf{I}_n \mp \mathbf{X}_1 (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} \mathbf{X}_1^T] \mathbf{e}_1 &= \mathbf{e}_1^T \{ \mathbf{I}_n \mp \mathbf{X}_1 [\mathbf{A} \mp \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{X}_1 \mathbf{A}] \mathbf{X}_1^T \} \mathbf{e}_1 \\ &= \mathbf{e}_1^T \{ \mathbf{I}_n \mp \mathbf{X}_1 \mathbf{A} \mathbf{X}_1^T + \mathbf{X}_1 \mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{X}_1 \mathbf{A} \mathbf{X}_1^T \} \mathbf{e}_1 \\ &= \mathbf{e}_1^T \{ \mathbf{I}_n \mp \mathbf{H}_1 [\mathbf{I}_n \mp (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{H}_1] \} \mathbf{e}_1 \\ &= \mathbf{e}_1^T \{ \mathbf{I}_n \mp \mathbf{H}_1 (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \} \mathbf{e}_1 \\ &= \mathbf{e}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{e}_1. \end{aligned}$$

Finally, the first expression for \mathbf{V}_N follows from the matrix inversion lemma, while the second is by definition.

Note that in each pair of equations for the updated quantities, the first equation is appropriate when $n \leq p$ while the second is for $n \geq p$. The sweep operator can be used in either case. For $n \leq p$, we have

$$\begin{aligned} \text{SWEEP}(1, \dots, p+n) \begin{bmatrix} \mathbf{X}^T \mathbf{X} & \mathbf{X}_1^T \\ \mp \mathbf{X}_1 & \mathbf{I}_n \end{bmatrix} &= \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A} \mp \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{X}_1 \mathbf{A} & -\mathbf{A} \mathbf{X}_1^T (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \\ \pm (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \mathbf{X}_1 \mathbf{A} & (\mathbf{I}_n \pm \mathbf{H}_1)^{-1} \end{bmatrix} \end{aligned}$$

and thus

$$\begin{aligned} \hat{\beta}_N &= \hat{\beta} \mp \mathbf{C} \mathbf{e}_1 \\ \text{RSS}_N &= \text{RSS} \pm \mathbf{e}_1^T \mathbf{E} \mathbf{e}_1 \\ \mathbf{V}_N &= \mathbf{B}. \end{aligned}$$

For $n \geq p$,

$$\begin{aligned} \text{SWEEP}(1, \dots, p) \begin{bmatrix} \mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1 & \mathbf{X}_1^T \mathbf{e}_1 \\ \pm \mathbf{e}_1^T \mathbf{X}_1 & \mathbf{e}_1^T \mathbf{e}_1 \end{bmatrix} &= \begin{bmatrix} \mathbf{U} & \mathbf{v} \\ \mathbf{w}^T & z \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} & (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} \mathbf{X}_1^T \mathbf{e}_1 \\ \mp \mathbf{e}_1^T \mathbf{X}_1 (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} & \mathbf{e}_1^T [\mathbf{I}_n \mp \mathbf{X}_1 (\mathbf{X}^T \mathbf{X} \pm \mathbf{X}_1^T \mathbf{X}_1)^{-1} \mathbf{X}_1^T] \mathbf{e}_1 \end{bmatrix} \end{aligned}$$

and thus

$$\begin{aligned} \hat{\beta}_N &= \hat{\beta} \pm \mathbf{v} \\ \text{RSS}_N &= \text{RSS} \pm z \\ \mathbf{V}_N &= \mathbf{U}. \end{aligned}$$

A.5.5. The General Form of the REG Command

The general form of the REG command is:

`REG(y,X,N,p,beta,Q,RSS,ni,t,e,lrsio[,V]).`

The input integer N can be either positive or negative. Its sign is used as a signal to REG, while its absolute value is the number of elements that the array \mathbf{y} has and the number of rows that the matrix \mathbf{X} has. Thus let $n=|N|$ and $i=1$ if N is positive and $i=0$ if N is negative.

The arguments \mathbf{y} ($n \times 1$), \mathbf{X} ($n \times p$), \mathbf{N} (integer), and p (integer) are input; the arguments \mathbf{t} ($p \times 1$), \mathbf{e} ($n \times 1$), and lrssio (real variable) are output; while \mathbf{beta} ($p \times 1$), \mathbf{Q} ($p \times p$), RSS (real variable), and \mathbf{ni} (integer variable) are both input and output. If $n \leq p$, then the $(n \times n)$ matrix \mathbf{V} is both input and output, while \mathbf{V} is ignored if $n > p$.

For the purposes of this discussion, we will attach a subscript I or O (denoting input and output) to any argument that is used for both input and output. There are two cases that REG handles, namely when the number of observations is greater than the number of parameters and when the number of observations is less than or equal to the number of parameters.

Number of Observations > Number of Parameters ($n > p$)

In this case, REG uses the following formulas:

$$\beta_O = \beta_I + i\mathbf{v}$$

$$\mathbf{Q}_O = \begin{cases} \mathbf{Q}_I + i\mathbf{X}^T\mathbf{X}, & \mathbf{ni}_I \neq 0 \\ i\mathbf{X}^T\mathbf{X}, & \mathbf{ni}_I = 0 \end{cases}$$

$$\text{RSS}_O = \begin{cases} \text{RSS}_I + is, & \mathbf{ni}_I \neq 0 \\ is, & \mathbf{ni}_I = 0 \end{cases}$$

$$\mathbf{ni}_O = \mathbf{ni}_I + in,$$

while

$$t(j) = \frac{v_j}{\sqrt{\frac{\text{RSS}_O}{\mathbf{ni}_O} \mathbf{U}_{jj}}}, \quad j = 1, \dots, p$$

$$\mathbf{e} = \mathbf{y} - \mathbf{X}\beta_O$$

$$\text{lrssio} = \log(\text{RSS}_I + i\mathbf{e}_1^T \mathbf{e}_1) - \log(\text{RSS}_O),$$

and \mathbf{U} , \mathbf{u} , \mathbf{v} , and s are calculated by

$$\text{SWEEP}(1, \dots, p) \begin{bmatrix} \mathbf{Q}_I + i\mathbf{X}^T\mathbf{X} & \mathbf{X}^T \mathbf{e}_1 \\ i\mathbf{e}_1^T \mathbf{X} & \mathbf{e}_1^T \mathbf{e}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{U} & \mathbf{v} \\ \mathbf{w}^T & s \end{bmatrix},$$

with $\mathbf{e}_1 = \mathbf{y} - \mathbf{X}\beta_I$.

Number of Observations \leq Number of Parameters ($n \leq p$)

In this case, we have:

$$\beta_O = \beta_I - iB_2e_1$$

$$Q_O = A_3$$

$$RSS_O = RSS_I + ie_1^T D_2 e_1$$

$$ni_O = ni_I + in$$

$$V_O = D_1,$$

while

$$t(j) = \frac{\beta_O(j)}{\sqrt{\frac{RSS_O}{ni_O - p}(A_2)_{jj}}}, \quad j = 1, \dots, p$$

$$e = Y - X\beta_O$$

$$lrsio = \log(RSS_O) - \log(RSS_I),$$

where $e_1 = y - X\beta_I$, and

$$\text{SWEEP}(1, \dots, p) \begin{bmatrix} Q_I & X^T \\ -iX & V_I \end{bmatrix} = \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix}$$

$$\text{SWEEP}(p+1, \dots, p+n) \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} = \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix}$$

$$\text{SWEEP}(1, \dots, p) \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix} = \begin{bmatrix} A_3 & B_3 \\ C_3 & D_3 \end{bmatrix}.$$

APPENDIX B

User's Guide to TIMESLAB

Throughout this book, the TIMESLAB program is used in examples and exercises. In this appendix and the next we provide the actual manual for the use of the program. This appendix is called the user's guide as it discusses installing the program, getting started in the process of using it, and its capabilities in general. Once the program has been installed, we strongly urge the reader to work through the "guided tour through TIMESLAB" given in Section B.3. In Appendix C, we provide detailed descriptions of each of the TIMESLAB commands.

TIMESLAB is called a laboratory because its aim is not so much to provide "canned" time series analysis programs (although it can be used that way), as it is to provide a setting in which various building blocks of time series analysis can be combined to study a series or method in depth. TIMESLAB is an integrated system in that it has its own editor, several high resolution graphics commands, the ability to read from and write to files, and the ability to issue DOS commands without exiting from the program.

Single commands exist to read or save data, to calculate a wide variety of quantities in both the time and frequency domain for either univariate or bivariate time series. Text and graphics output can be merged on a printer, and graphics screens can be saved onto files for later display. There is also the ability to process files called macro files that contain a series of TIMESLAB commands. This ability means that TIMESLAB can be used as a time series analysis programming language.

B.1. Hardware and Software Requirements

To use TIMESLAB, one needs:

1. An IBM PC, PC/XT, PC/AT or any personal computer that is compatible with the IBM family of PC's.

2. 640K RAM.
3. The IBM Color Graphics Adapter (CGA) or Enhanced Graphics Adapter (EGA) or any adapter compatible with these.
4. A monitor that can perform pixel graphics and is compatible with the graphics adapter that is installed in the PC.
5. Either two floppy disk drives or a hard disk.
6. An 8087 (or 80287 if one has a PC/AT or AT compatible) numeric coprocessor.
7. For printed output, any nine pin dot matrix printer that recognizes the printer commands of the IBM Graphics Printer, any 24 pin dot matrix printer compatible with a Toshiba 321 Printer, or any laser printer compatible with the Hewlett-Packard Laserjet or Laserjet Plus Printer.
8. Version 2.0 or later of DOS.
9. When the computer is started by turning it on (this is referred to as a "cold boot") or restarted by issuing the Ctrl-Alt-Del "warm boot" from the keyboard, two files must be on the root directory of the "booting" device (i.e., the hard disk if there is one or a floppy which has been formatted with the /S option in the DOS FORMAT command): (1) the file called ANSI.SYS which comes with DOS, and (2) a file called CONFIG.SYS containing the DOS command

DEVICE=ANSI.SYS.

The presence of these files makes it possible for TIMESLAB to use the colors (or shades of green or amber on a monochrome monitor) in the text mode. No other software is required as TIMESLAB is designed as an integrated system; that is, it has its own full screen editor, its own graphics, and its own communications to the printer. Any editor or word processor can be invoked from within TIMESLAB by issuing the DOS command first. This command establishes a modified DOS prompt from which any command issuable from DOS can be issued without affecting what is being done in TIMESLAB.

B.2. Installing and Starting TIMESLAB

Everything that is needed to use TIMESLAB is contained on the two diskettes accompanying this book. TIMESLAB is not copy protected in any way.

Diskette 1

The only file on this diskette is TIMESLAB.EXE which is the executable file containing TIMESLAB.

Diskette 2

This diskette contains the following files:

1. TSLABHLP: This file contains the text of the help messages in TIMESLAB. Note that TSLABHLP is not printable.
2. HELP: This is a printable version of the help file.
3. TSLABW1: Any file having this name can be formed by the user and its contents will be displayed on what is called a "second welcoming screen." See below for more details on the purpose of this file.
4. All of the data sets referred to in this book. These are files having extension .DAT. The first part of their name indicates which of the data sets discussed in the book that they are. See Appendix D for a listing of each of the data sets.
5. All of the macros discussed in the book. These are the files having extension .MAC. See Appendix E for an index to the macros.
6. A few files having extension .SCN. These files contain graphics screen images. They can be "re-screened" using the command RESCREEN.
7. A file called EXAMPLE.NTE. This file is an example of what is called a notes file and is used with the NOTES command.

The TSLABW1 File

When TIMESLAB is started, it displays a welcoming screen and waits for the user to strike a key to continue. If there is a file called TSLABW1 on the current subdirectory of the default disk drive, then a second welcoming screen is displayed containing the contents of TSLABW1. If not, then the second welcoming screen is not displayed. Users can form their own file called TSLABW1 containing, for example, messages to someone else using TIMESLAB. An instructor for a time series course could create a file containing notes to the students taking the course.

Installing TIMESLAB

The first thing to be done is to make backup copies of the distribution diskettes. See the beginning of any DOS manual for a description of how to do this. Once these copies are made, the method in which TIMESLAB should be installed depends on whether the user has a hard disk or not.

There are three basic ways to use TIMESLAB:

1. On a two-floppy system.
2. On an unshared hard disk system.
3. On a shared hard disk system.

Two-Floppy System

In this situation two diskettes should be formed, one called the A floppy which contains a copy of COMMAND.COM (so that the DOS command will work) as well as TSLABW1, TSLABHLP, and any .MAC, .SCN, .NTE, and .DAT files to be used, and the other floppy called the B floppy which contains only TIMESLAB.EXE. Once this is done, with the DOS prompt being **A>**, the command

```
B:TIMESLAB
```

will start TIMESLAB. Setting things up this way assures that there is sufficient room on the default floppy (i.e., floppy drive A) to receive data, macro, and screen files. Once TIMESLAB has been started, the program diskette in drive B can be removed and both drives can be used to contain data sets, macros, and screen images.

Unshared Hard Disk System

In this situation, a subdirectory should be formed containing everything on the distribution diskettes. Then TIMESLAB can be invoked from a prompt designating the hard disk by just saying its name.

Shared Hard Disk System

In some cases readers will be using a copy of TIMESLAB residing on a hard disk that someone else put there. In this situation, a floppy can be used to receive any files such as macros, data files, or screens produced during a TIMESLAB session. Thus a floppy should be created containing anything that is needed (it doesn't need TIMESLAB.EXE or COMMAND.COM), the DOS

prompt should be set to **A>**, and then the command (assuming that **TIMESLAB.EXE** is on hard disk drive **C**)

```
C:TIMESLAB
```

will start **TIMESLAB**.

Getting Colors Other Than Black and White

In order to get colors other than black and white in the text mode of **TIMESLAB**, the PC being used must have been booted with a file called **CONFIG.SYS** residing on the root directory of the booting disk drive and **CONFIG.SYS** must contain the line

```
DEVICE=ANSI.SYS.
```

Further, the file called **ANSI.SYS** that comes with versions 2.0 and later of **DOS** must also be on the root directory of the booting disk. If this is not the case, no colors will be produced in the text mode and strange characters will appear on the screen whenever **TIMESLAB** switches from graphics to text mode. Note that these characters will appear on the printer sometimes in any event.

B.3. A Guided Tour Through TIMESLAB

To get an idea of how **TIMESLAB** works, we suggest that you work through the following tour of the program. This tour only considers a small fraction of the capabilities of the program but should be useful in getting started. For the tour to be successful, the following must be true:

1. You are using an IBM-compatible PC or PC/XT or PC/AT with 640K RAM that has a color graphics adapter or enhanced graphics adapter and either a color monitor or monochrome monitor that can perform pixel graphics.
2. You have booted your PC with a file called **CONFIG.SYS** (that you created) and a file called **ANSI.SYS** (that comes with **DOS**) on the root directory of the disk that you booted from, and included in the file **CONFIG.SYS** is the line **DEVICE=ANSI.SYS**.
3. You are logged on to the subdirectory that contains all of the files on distribution diskette number 2 and can access the file **TIMESLAB.EXE**. If you are using floppys this usually means that diskette 2 is in **A** and diskette 1 is in **B**.

The Two Welcoming Screens

If these steps have been performed, then it is time to start TIMESLAB. Start the program according to the instructions in Section B.2. The first thing that you will see is a screen welcoming you to the program. On a color monitor this screen will have a blue background with a primarily yellow foreground. On a monochrome monitor, these colors will be represented by different shades of the color used by the monitor. If you did not get these colors, then you did not boot your PC with the files called ANSI.SYS and CONFIG.SYS set up properly as described above. If this is the case, strike the enter key twice, type the command QUIT, and then strike the enter key again. This returns you to DOS and you can review Section B.2 on how to set up your files properly. If you did get the colors and you don't like them, you can change them later by using the TEXTCOLOR command.

On the bottom of the screen is a message directing you to strike any key to continue. Do this. The next screen that you see is an optional second welcoming screen. At the top of the screen is a heading telling you that you are looking at a second welcoming screen, and at the bottom is a message saying to strike any key to continue. The contents of this screen are contained in the file TSLABW1 that came on distribution diskette 2. If you did not get this screen, then you are not logged onto a subdirectory containing the files that came on diskette 2, and this tour through TIMESLAB will not work properly. If this is the case, then you are probably looking at a screen with a question mark at the bottom. Type the command QUIT and strike the enter key. This should return you to DOS and you can return to Section B.2 to set up your files correctly. If you did get the second welcoming screen, strike any key and the second welcoming screen will disappear and a list of commands and the TIMESLAB prompt will appear on the screen. The PC will beep to indicate that it is now ready for you to enter commands.

The List of Commands and the TIMESLAB Prompt

You should now be looking at a screen that has a list of the 150 or so commands that are available. At the bottom of this list are two lines that tell you how to get information about the commands and then finally a question mark with the cursor next to it. This question mark is called the TIMESLAB prompt and is analogous to the DOS prompt (such as A>). You can get this same screen any time the prompt is on the screen (in the text mode, see below) by entering the command HELP. Note that entering a command means to type the command and then strike the enter key (also called a carriage return). In the rest of this tour through TIMESLAB, things that you will type will be represented by typewriter font (*like this*). We will always represent the command names by typing them in uppercase, although you can enter them in upper or lowercase. You can leave TIMESLAB and return to DOS by entering the command QUIT at any time.

The Text and Graphics Modes

The monitor that is connected to an IBM PC and operated by a color graphics adapter can operate in two "high resolution" modes; (1) a text mode which consists of 25 rows of 80 character locations, and (2) a graphics mode which consists of 200 rows of 640 pixel locations. A character is actually made up of an 8×8 pixel box, but the individual pixels are only addressable in the graphics mode. On a PC equipped with the Enhanced Graphics Adapter (EGA), there are more pixels on the screen than the 640×200 for the CGA but TIMESLAB does not make use of this extra resolution.

The two modes are different in the following ways:

1. High resolution graphics can only be formed in the graphics mode. If one of the graphics commands is issued from the text mode, then TIMESLAB (1) switches to the graphics mode (which erases whatever is on the screen), (2) produces the plot, and (3) waits for the user to signal that he or she is finished viewing the plot, at which point it switches back to the text mode (which erases the plot).
2. In the text mode, the TIMESLAB prompt moves down the screen until it reaches the bottom at which point the entire screen is scrolled up. In the graphics mode, the prompt is reissued at the top of the screen so that it does not overwrite any graphics that may be on the screen.

Thus unless one wants to be able to have more than one graph on the screen at the same time, it is best to leave TIMESLAB in the text mode.

When TIMESLAB is started, it is in the text mode. To switch from the text mode to the graphics mode, one issues the command `PLOTON`. Do this now; that is, type `PLOTON` and then strike the enter key. If you have a CGA, the screen should be black with a brown prompt. If you have an EGA, the screen should be brown with a green prompt. In the graphics mode you can get only two colors on the screen at once, one of which is called the background and another called the foreground. If you have a CGA, enter the command `COLOR(2)`, and if you have an EGA, enter the commands `COLOR(256)` and `COLOR(0)`. Now the prompt should be green for either adapter with a black background. You can experiment later with setting graphics colors with the `COLOR` command. Now enter the commands

```
x=WN(0,100)
PLOT(x)
```

You should have a graph of a white noise time series on the screen and the prompt still at the top of the screen. In the graphics mode, it is always issued at the top of the screen. Otherwise it would interfere with any graphs that are on the screen.

To switch back to the text mode, the `PLOTOFF` command is used. Do this now. Now the screen should be blue with a white foreground. These colors can be changed by using the `TEXTCOLOR` command. Enter the command `TEXTCOLOR(7,1)`. This makes the background white and the foreground blue. To get back to the original text colors, enter the command `TEXTCOLOR(1,7)`.

Getting Help

The TIMESLAB help system can be accessed at any time that the program is in the text mode and prompting you for a command. Enter `HELP` now. Now enter `HELP(wn)`. You should have a short description of the `WN` command on the screen. Each command is more fully described in Appendix C and somewhere else in the book where the method used in the command is described (see Appendix E for an index of commands). If you got a message saying that TIMESLAB couldn't find the file `TSLABHLP`, then you haven't set up your files properly and you should leave TIMESLAB and do so.

Now enter `HELP` again. This time notice the message at the bottom of the screen saying that you can get a list of general help areas by entering the command `HELP(areas)`. Do this now. Notice the list of areas. Now enter the command `HELP(matrices)`. This will give a list of the commands that can be used to study matrices in TIMESLAB. Now enter `HELP(estimation)`. This gives a list of commands for estimation.

Data Types

Now we will define some variables. Enter the following commands one at a time. Make sure they are entered exactly as given. In particular, make sure that the decimal point is included in third and fourth commands. If you make a mistake in entering a command, TIMESLAB will alert you to the fact and you should be able to just enter it again.

```
k=10
m=5
c=1.
d=5.
twopi=2.*pi
fname='air.dat'
READ(air,x,n)
fni='wn#k#'
t=<1,2,3,4,5>
LABEL(t)='The Numbers 1 Through 5'
y=WT(12345,100)
w=COS(100,1,10)
z=y+w
INFO
```

You should see the following on your screen:


```

Number of arrays and Free Elements:      5   9551
NAME          LENGTH LABEL
-----
x             144 International Airline Passengers
t              5 The Numbers 1 Through 5
y            100 normal white noise series
w            100 cosine curve
z            100 sum of two arrays
-----
k              =          10 m              =          5
n              =          144
c              =      1.00000000 d          =      5.00000000
twopi          =      6.28318500
fname          = air.dat          fn1          = wn10

```

You have defined the integers **k** and **m**, the real scalars **c**, **d**, and **twopi**, the character variables **fname** and **fn1**, and the four arrays **y**, **z**, **w**, and **t**. The **READ** command resulted in the integer **n** and the array **x** being defined as well. The **INFO** command will tell you all of the variables that you have defined at any time. The basic rules are as follows (much more detail is given in Section B.4):

1. Variable names can have as many as 15 characters, you can have as many as 60 of each of the four types, and the total number of elements for all of your arrays must be less than or equal to 10,000 (**INFO** tells you how many "free elements" you have left).
2. Each array has a 40-character "label" associated with it. This label is used at the top of lists and plots of the array. When you read a data set from a file (via the **READ** command as you entered above for the International Airline Data on the file **AIR.DAT**), the label that is in the file (see Section B.4.8) is the one that **TIMESLAB** uses. When an array is created by a command (such as the command **y=WN(12345,100)** above), **TIMESLAB** creates its own (often not very informative) label for it (see the labels for the arrays **y**, **z**, and **w**). This label can be changed by the **LABEL** command (see the one that you entered above for **t** and the result in the output of **INFO**).
3. **TIMESLAB** could tell that you wanted **k** and **m** to be integers and **c** and **d** to be reals because of the absence or presence of a decimal point. On the other hand, when **TIMESLAB** is expecting a real as an input argument, and you want to enter a real that is a whole number, you need not enter the decimal point (the first argument in the **WN** command was a real). There were two other things to notice in the example above: (1) the value of π is always available in the variable **pi** (therefore don't define a variable called **pi**), and (2) in the line defining **fn1**, the value of **k** is inserted into the character variable. This is often a useful capability.

Listing Variables

To list the values of arrays or other types of variables, we have the **LIST** command. Enter the command **LIST(k,fn1,c)**. You should see the values of these variables. Now enter **LIST(x,50)**. This will display the first 50 elements of the airline data (as well as its label). Now enter **LIST(y,100,10,10f6.2)**. This will list the normal white noise array with 10 elements per line in the Fortran format **10f6.2**. Note that you can't list both scalars and arrays in one command.

Graphics

There are several graphics commands in **TIMESLAB**. The simplest is the **PLOT** command, which does a simple *Y* versus *X* plot. There are several forms of this command (see Appendix C). We illustrate the **PLOT** command by plotting the airline data. Enter the command **PLOT(x,n)**. You should see a graph of the data (on the vertical axis) versus the index 1 through 144 on the horizontal axis. At the bottom of the screen is what is called the graphics menu, giving you a choice of several keys that you can strike. (Do not strike any of these keys now.) The graphics menu is described in detail in Section B.5.3. Note the following:

1. The label for this array is at the top of the screen.
2. There are ten tic marks on each axis.
3. The ends of the axes correspond to the largest and smallest values of the arrays being plotted.
4. The numerical labels on the tic marks are "ugly," particularly those on the horizontal axis which we would like to be whole numbers. We will correct this in a minute.
5. There are no descriptive labels below the horizontal axis or to the left of the vertical axis. Such labels can be added to a plot either using the **FIND** utility (see Section B.5.5) or using the **LABEL** command.

We will next redraw the plot that is on your screen. Notice that the values on the vertical axis range from 104 to 622, while those on the horizontal axis range from 1 to 144. There is another form of the **PLOT** command where you can specify the values that you want at the ends of the axes. Since there are ten tic marks, if we put 100 at the bottom of the vertical axis and 700 at the top, then the plot will fit and the tic mark labels will be 100, 160, 220, and so on. Similarly, if we specify that the ends of the horizontal axis are to be 0 and 150, then the labels will be 0, 15, 30, and so on (except that only every other

tic mark gets labeled). Now strike any key. Enter the command

```
PLOT(x,n,0,150,100,700)
```

and see how much better the tic mark labels are. This also makes it so that the plot moves away from the axes slightly. Note that the size of plots, as well as the number of tic marks and the number of digits in the numerical labels placed at the tic marks, can be specified using the `PLOTSIZE` command. Certain standard sizes and locations of plots have been defined in `TIMESLAB`. To illustrate this, enter the commands `PLOTSIZE(4)` and `PLOT(x,n,0,150,100,700)`, and note that the graph is now in the lower right quarter of the screen. Now enter the command `PLOTSIZE(0)`, strike the enter key again, and go on to the next section.

Using Macros

The ability of `TIMESLAB` to read commands from a file is what makes it so powerful. Such a file is called a macro. To illustrate this, we will use the `WNTTEST` macro (it resides in the file `WNTTEST.MAC` that came on the second distribution diskette and is discussed in Example 1.5) to determine if there is significant autocorrelation in the airline data. First, find the log of the data by entering the command `x=LOGE(x,n)` (the reason for taking the log is discussed in Section 1.5). Issue the command `MACRO(wntest)`. You should see several lines on the screen starting with two semicolons and containing a description of what the macro does and what you have to define in order to use it. This is called the "prolog" of the macro and all of the macros that come with `TIMESLAB` have such a prolog. The last line should say `PAUSE... STRIKE F9 TO BREAK, ANYTHING ELSE TO GO ON`. The `WNTTEST` macro expects that you have defined three variables: (1) an array `x` containing data, (2) an integer `n` containing the number of elements of the array `x` to analyze, and (3) an integer `m` containing the number of correlations to use (don't worry about what this means for now). We defined `x` and `n` when we read the airline data, and we defined `m` to be 5, which is too small a value to be used in this context. Thus strike the F9 function key to "break out" of the macro. The `TIMESLAB` prompt now appears. Enter the command `m=30` (no decimal point since this is an integer). Now restart the macro (right where it left off) by entering the command `MACRO`. Now `TIMESLAB` starts executing one command after another from the `WNTTEST.MAC` file (the commands should go by quickly). Finally, the macro will pause with a graph on the screen with two horizontal lines and another curve above them. If there were no correlation in the data, this curve should be between the horizontal lines. Now strike any key (not F1, F3, F5, or F9 please), and the macro will resume execution, and finally you should see another graphics screen with three diagonal lines and another curve above them. Again, if there were no correlation in the data, this curve should be between the diagonal lines. Now strike any key and note that the macro

has ended because the TIMESLAB prompt has reappeared.

Another Macro

Another interesting macro is the one called **COS.MAC** (see Example 1.4). This macro forms four cosine curves of various periods and amplitudes, plots all four on the screen at once, then adds them together and plots the resulting data and periodogram on the screen at the same time. To see this, enter the command **MACRO(cos)**. You will see the prolog and the message to strike any key to continue. Do this. Now you will see the lines of the macro at the top of a graphics screen as they are being executed, and finally the four cosine curves are plotted on the screen together with the graphics menu. Strike any key (except F1, F3, F5, or F9), and you will see the data and the periodogram plotted on the screen. Note that the periodogram has peaks at the four frequencies corresponding to the periods of the four component cosine curves. Now strike any key and the TIMESLAB prompt should reappear in the text mode.

The FIND Utility

An important feature of the graphics in TIMESLAB is the presence of an interactive locator/editor from any graphics screen. To illustrate this, enter the commands **PLOTSIZE(0)** (which resets the size parameters) and **PLOT(x,n)**. You now should have a plot of the airline data that fills much of the screen. Part of the graphics menu is the option of striking F1 to enter the **FIND** utility. Do this. A box (called a "locator") should appear in the upper right corner as well as the values of *x* and *y* where the locator is. Note that at the bottom of the screen is the **FIND** menu. This menu tells you that you can leave the **FIND** mode (don't do this now) by striking Alt-F10 (i.e., hold down the Alt key while striking the F10 key, and then release the Alt key), or you can get directions on how to use **FIND** by striking Alt-F1 (don't do this either). Now strike the key labeled 2 on the numeric keypad (not the one along the top of the keyboard). The locator should move down a little and the values of *x* and *y* in the upper right-hand corner of the screen should adjust to the move. If instead of the locator moving, a 2 appeared on the screen, then strike the Num Lock key and try it again. Note that you can move the locator in any direction by using the corresponding key on the numeric keypad (e.g., striking 7 moves it up and left, striking 1 moves it down and left, and so on). Now strike Ctrl-4; that is, hold down the Ctrl key while striking the 4 on the numeric keypad. The locator should have moved all the way to the vertical axis. Note that the numbers must be entered from the numeric keypad.

Now type your name on the screen. Note that any of the 256 characters can be placed anywhere on the screen. If a character has a key, just strike its key. If not, then you must enter its ASCII code by holding down the Alt key, typing the ASCII code for the character, and then releasing the Alt key. When the Alt key is released, the character will appear. To illustrate this, we will

place an α on the screen. We need to find the ASCII code for an α . Strike Alt-F1 and notice that you now have a help screen. Note that there are a variety of other things mentioned on this help screen, including directions for drawing lines or rectangles. At the bottom of the screen is a message saying to strike F1 to get a list of ASCII codes. Do this. Look up α in the table that is now on the screen. It should be character number 224. Now there is a message at the bottom of the screen saying to strike any key to return to FIND. Do this. Now strike Alt-224 (i.e., hold down the Alt key, type 224 on the numeric keypad, and then release the Alt key) and see how the α appears on the screen. There are a number of other features of FIND (see Section B.5.5), but for now we leave it. Strike Alt-F10 and note that you are back to the graphics menu. Strike any key and we're back at the TIMESLAB prompt.

Retrieving Graphics Screens

To illustrate how graphics screens that have been previously constructed and saved can be instantly redisplayed, enter the command

```
RESCREEN(smooth.scn)
```

This should place a copy of Figure 1.10 of the book on your screen. Now strike a key to get rid of this plot and then enter the command

```
RESCREEN(wind1.scn,wind2.scn,wind3.scn,1)
```

and see what you get. Now strike the enter key to return to the TIMESLAB prompt and go on to the next section.

The DOS Mode

It is possible to issue DOS commands while within TIMESLAB. To illustrate this, enter the command DOS. A prompt will appear which is a combination of the TIMESLAB prompt and the DOS prompt. There is also a message telling you to enter the command EXIT (in either upper or lowercase) in order to get back to the TIMESLAB prompt. Now issue the command

```
DIR *.DAT
```

and you should get a list of all the data files that were on the distribution diskette. Now enter EXIT and the ordinary TIMESLAB prompt should reappear.

Two Fun Macros

To finish this tour, we will use two other macros: (1) **CIRCLE**, and (2) **ODE** (see Examples B.5 and B.4 respectively). Enter **MACRO(circle,start)** (the second argument tells **TIMESLAB** to begin execution of the macro at the line **;start** in the **CIRCLE** macro—this skips the prolog). You will get a solid circle on the screen. Now strike the enter key and enter the command **MACRO(ode,start)**. What do you get?

Exiting from TIMESLAB

To leave **TIMESLAB** and return to DOS, enter the command **QUIT**. Note that none of the variables that you have defined will be available at your next session unless you save them with the **SAVE** command.

The Next Step in Learning TIMESLAB

At this point, you should have a rough idea of some of the basic features of **TIMESLAB**. The next step is to learn more commands, to learn how to use the **TIMESLAB** editor (see Section B.6) so that you can form your own macros (see Section B.7 for details on macros) and data sets (see Section B.4.8 for details about data sets), and to study the rest of this appendix for more details about the general features of the program.

B.4. Overview of TIMESLAB Capabilities

In this section we describe the general features of **TIMESLAB**. We begin by describing the five types of data recognized by **TIMESLAB** and the five types of commands that it has.

B.4.1. Data Types

There are five types of data recognized by **TIMESLAB**.

Immediate Values

These are integers such as 2, 7, and 777, real numbers like 3.88, -75.90, and 3.1415926, and character values like 'character', 'obs.dat', 'hello mom'. Note that character immediate values are just strings of characters enclosed in apostrophes. When used as arguments in a command, immediate character values need not have the apostrophes. Integers are between -32,767 and 32,767, reals between around 10^{-37} and 10^{37} , while characters can have at most 15 characters.

Character Variables

These are variables whose names have at most 15 characters and whose "value" is a string of characters containing at most 15 characters. Character variables are formed either as output of a command or by a simple assignment statement of the form

```
name='string'
```

which is a type 2 command and is described below. Note that the command above sets a character variable equal to an immediate character value.

Integer Variables

These are variables whose names have at most 15 characters and whose value can be any valid two-byte integer ($-32,767$ to $32,767$). For example, in the command `n=3`, `n` is an integer variable while `3` is an immediate integer value.

Real Variables

These are variables whose names have at most 15 characters and whose value can be any valid four-byte real value (approximately 10^{-38} to 10^{38}). There are two real scalar variable names that are always available, namely `pi` and `exp` which have the values π and e . Note that one can use an element of a real array as a real scalar by using the notation `x[i]`. For example one can say `x[i]=y[10]+z[j]` where `x`, `y`, and `z` are real arrays.

Real Arrays

These are variables whose names can have at most 15 characters and which contain sets of real numbers. Real arrays are the basic objects manipulated by TIMESLAB; a time series data set is stored in an array, values of a correlogram or spectral density are arrays, and matrices are stored in arrays. Arrays can be formed either by a command of the form

```
A=<1,2,3,4,5,6>
```

or as output of a command. Matrices are stored by column as in Fortran. For example, the array `A` above can be thought of as a 1×6 , 2×3 , 3×2 , or 6×1 matrix. If in a command, `A` is said to be 3×2 , then TIMESLAB treats it as

$$A = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

B.4.2. Command Types

The commands in TIMESLAB can be classified in terms of either their form or their utility. The structure of the program and the error messages it generates refer to the classification in terms of their form.

Type 1 Commands

These commands are of the form

```
command name(list of arguments)
```

For example,

```
PLOT(x,n), TEXTCOLOR(0,2), and READ(air.dat,x,n)
```

are type 1 commands.

Type 2 Commands

These commands consist of either an assignment statement such as `n=10` or operations such as `z=x+y` or `z=x+10`. The operations that are available are addition (+), subtraction (-), multiplication (*), division (/), exponentiation (^), and negative exponentiation (^ -). These operations can be applied to integers, reals, and arrays, and several mixed type operations are available; for example, a real scalar `c` be subtracted from each element of the array `X` by the command `Y=X-c`. See Section B.4.3 for a complete description of the type 2 commands.

Type 3 Commands

These commands are of the form

```
variable name=command name(list of arguments)
```

For example, the commands

```
x=WN(seed,n) and y=SORT(x,n)
```

are type 3 commands.

Type 4 Commands

Examples of these commands are

$$\mathbf{x}=\langle 1,2,3,4,5\rangle \quad \text{and} \quad \mathbf{x}=\langle \mathbf{x},5.43,\mathbf{z},\mathbf{y},6.2\rangle,$$

where any number of scalars or arrays can be concatenated. This is how vertical and horizontal concatenation of matrices is done. For example if $\mathbf{A}=\langle 1,2,3,4,5,6\rangle$, $\mathbf{B}=\langle 11,12,13,14\rangle$, and $\mathbf{C}=\langle \mathbf{A},\mathbf{B}\rangle$, and one of the matrix commands (e.g., **MMULT**) is told that \mathbf{C} is a (2×5) matrix, it will treat it as

$$\mathbf{C} = \begin{bmatrix} 1 & 3 & 5 & 11 & 13 \\ 2 & 4 & 6 & 12 & 14 \end{bmatrix}.$$

However, if \mathbf{A} and \mathbf{B} are treated as

$$\mathbf{A} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 11 & 13 \\ 12 & 14 \end{bmatrix},$$

and it is desired to vertically concatenate them, one would have to do the following steps:

```
A=TRANS(A,3,2)
B=TRANS(B,2,2)
C=<A,B>
C=TRANS(C,2,5)
```

Type 5 Commands

These commands have no arguments and are given by merely entering their name. The **HELP** command with no argument is a type 5 command, while the **HELP** command with an argument is a type 1 command.

B.4.3. Simple Assignments and Arithmetic Operations

The type 2 command in **TIMESLAB** is used to do assignments and arithmetic operations. The general form of the assignment statement can be expressed as (the equal sign can be preceded by a colon also)

```
out=e
```

where:

- out - Either a variable name or an element of an array ($\mathbf{x}[i]$ for example) which will receive the result of the assignment.

- e - Either an immediate integer, real, or character, a variable name, or an element of an array. The presence or absence of a decimal point determines whether an immediate value is taken to be real or integer.

Some examples include (the code including semicolons and after are comments):

```

n=1           ;integer since no decimal point
c=3.          ;real
x=<1,2,3,4>    ;an array (this is a Type 4 command, not a Type 2)
y=x           ;assign the array x to the array y
fname='air.dat' ;character variable
fni='wn#n#'    ;insert the value of a variable into a character
x[2]=x[3]     ;operate on elements of an array
n:=c          ;convert the real c to an integer n

```

The following rules are used to determine how the assignment is done:

1. If there is no colon preceding the equal sign, then:
 - a. If the left-hand side is the element of an array, then the value of the right-hand side is assigned to that element of the array unless the right-hand side is itself an array in which case the command aborts.
 - b. If the left-hand side is not an element of an array, then a variable having name specified by out is created having the same type and value as the right-hand side.
2. If there is a colon preceding the equal sign, then:
 - a. If the left-hand side is an element of an array, then the right-hand side is truncated and assigned to the specified element of the array unless the right-hand side is itself an array in which case the command aborts.
 - b. If the right-hand side is an integer, then a real variable is formed containing its value unless the left-hand side is an element of an array in which case it receives the value of the right-hand side.
 - c. If the right-hand side is a real, then its value is truncated and an integer variable is formed unless the left-hand side is an element of an array.

Arithmetic Operations

The arithmetic operations that can be performed in TIMESLAB can be expressed in the general form (the equal sign can be preceded by a colon)

$$\text{out} = e_1 \otimes e_2 \otimes \cdots \otimes e_k$$

where:

- out** - Either a variable name or an element of an array (**x[i]** for example) which will receive the result of the operation.
- e_i** - Either an immediate value, a variable name, an element of an array, or an arithmetic operation itself within curly brackets (**{x+y}** for example).
- ⊗** - One of the operators **+**, **-**, *****, **/**, **^**, or **^-** for addition, subtraction, multiplication, division, exponentiation, and negative exponentiation.

If the equal sign is preceded by a colon, this indicates that the user wants to either truncate or change the type of the right-hand side before assigning its value to the variable or array element specified by **out** (see below). Some examples of arithmetic operations are:

```
j=2                      ;operations on integers
k=3
n=2*j+k
x=LINE(201,-1.01,.01)    ;this is -1, -.99,..., 0,..., .99, 1.00
x=4*x                    ;now -4 to 4
fx=exp^(-x^2/2)/(2*pi)^.5 ;this is normal pdf at -4 to 4
```

How Expressions Are Evaluated

When TIMESLAB encounters an arithmetic operation of the above form, it first evaluates the right-hand side, including determining its type (integer, real, or array) and value, and then places the result into the variable or array element specified by **out**. It first evaluates any expressions that are within curly brackets (working from left to right and always doing the innermost pair of brackets first if there are any nested brackets) and creates variables containing their value. Note that any variables created by such evaluations are deleted before the execution of the entire command is finished. This means that the user never sees these variables—for example they would not appear on the lists of variables if the **INFO** command were issued. If creating one of these variables would cause either the 60 variables of one type limit or the 10,000

array elements limit, TIMESLAB issues an error message to this effect, and the command aborts.

To evaluate an expression within curly brackets or to evaluate the overall expression once all bracketed expressions have been evaluated, TIMESLAB uses the following rules:

1. All exponentiations and negative exponentiations are evaluated, then all multiplications and divisions are evaluated, and finally all additions and subtractions are evaluated. In each case, the first operation from the left within the three types is the one evaluated next.
2. The result of each such binary operation is placed into a temporary variable of a type determined by the type of the two quantities being operated upon. A quantity can be either an immediate or variable integer, an immediate or variable real, or an array. In this case TIMESLAB determines whether an immediate value is real or integer by the presence or absence of a decimal point. The type of the result of a binary operation is then determined by the following rules:
 - a. If the two quantities are both integers, then the result is taken to be an integer unless the operation is to raise an integer to a negative power, in which case the result is taken to be real.
 - b. If the two quantities are both reals, then the result is taken to be real.
 - c. If one of the quantities is real and the other is an integer, then the result is taken to be real.
 - d. If both quantities are arrays, then the operation is applied on corresponding elements of the two arrays, and the result is taken to be an array of length equal to the shorter of the two arrays if they are not of the same length.
 - e. If one of the quantities is an array and the other is either an integer or a real, then the result is taken to be an array.

Once the type and value of the right-hand side of the equal sign are determined, then the same rules that were used in the simple assignment statement form of the type 2 command are used to do the assignment part of the arithmetic operation form.

Numerical and Syntax Errors

The following numerical errors will cause a command to abort with a message being displayed:

1. Division by zero.
2. An integer or real zero to a negative power.
3. A negative real (including an element of an array) to a real power.
4. An operation on reals that results in a value that is greater in absolute value than the largest four-byte real number (approximately 10^{38}).
5. An operation on integers that results in a value that is greater in absolute value than 32,767.

The following conditions will cause a command to abort with the message 'Syntax Error or Undefined Term' being displayed:

1. An undefined variable being used on the right-hand side of the equal sign.
2. A quantity (either an immediate value or variable name) more than 15 characters long.
3. An illegal element of an array being used, for example, $\mathbf{x}[\mathbf{i}]$ where \mathbf{i} is undefined or less than 1 or greater than the length of \mathbf{x} .
4. Two operations (other than \wedge) occurring next to each other. Thus $\mathbf{c}=-2.*-3.$ is illegal while $\mathbf{c}=-2.*\{-3.\}$ is legal.
5. An unmatched curly bracket.
6. Nothing to the left of the equal sign.
7. Trying to assign an array to an element of an array.

B.4.4. Listing and Deleting Variables

TIMESLAB has room for 60 of each of integer, real, and character variables as well as for 60 arrays having a total of 10,000 elements. Each variable or array has a name attached to it. Each array also has a label having as many as 40 characters. To get a list of all variables and arrays, one need only enter the command INFO. This command also gives the length of each defined array, its label, and how many free elements are left of the original 10,000 elements.

The various forms of the **LIST** command can be used to display the values of variables that have been defined, while the various forms of the **CLEAN** command can be used to delete variables that have been defined.

B.4.5. Cases and Types

Command names and DOS file names and device identifiers can be given in either upper or lowercase while all other names are case sensitive; for example, if an array called **x** has been defined, then **TIMESLAB** won't recognize the array **X** in **PLOT(X,n)**. This means that one can use the same name in upper and lower case for two different variables if desired.

Note that the same name cannot be given to variables of different type.

The only time that a "whole number" real immediate value (e.g., 23.) need have the decimal point attached is in a type 2 assignment statement. For example, the command **x=10** will cause **TIMESLAB** to form an integer variable called **x** having the value 10. On the other hand, any time a real number is expected as an input argument in a command (as in **seed** in the command **X=WN(seed,n)**), and the desired value is a whole real number, the decimal point need not be included. Thus **X=WN(12345,n)** is a valid command even though the **WN** command expects a real scalar as the first argument.

B.4.6. Error Checking

TIMESLAB does extensive error checking. When a command is received, it first attempts to determine its type. If the command fits none of the five types, an "unrecognizable command type" message is displayed and the prompt reappears. The most common way this happens is for the user to leave off a parenthesis.

Once the command type has been determined, the command name is checked against an internal list of available commands of that type. If it doesn't exist, a message to that effect is displayed and the prompt reappears. For types 2 and 4 this doesn't apply since there is no command name associated with these types.

Once **TIMESLAB** determines that a legitimate command has been issued, it checks any input arguments for having been defined. If a needed argument hasn't been defined, the program issues an appropriate message and reissues the prompt. This often happens because a variable name has been misspelled. Again, any argument can have at most 15 characters, there can be at most 21 arguments, and a command can have at most 72 characters.

TIMESLAB also checks many arguments for being in legal ranges; for example, in **PLOT(X,n)**, **TIMESLAB** checks whether previously an array **X** having at least **n** elements has been defined.

Whenever **TIMESLAB** receives a command that requires it to use some of the computer's memory as a workspace, it checks to make sure that enough workspace is available. An example of this is given by the command

G=TOEPL(rho,R0,n).

This command is used to form an ($n \times n$) Toeplitz matrix. TIMESLAB forms the matrix in a workspace and then copies it into the area of memory where arrays are stored. The workspace that TIMESLAB uses is restricted to 10,000 elements. Thus if a user were to inadvertently try to form a 200×200 matrix, TIMESLAB would issue a message that there is not enough workspace and the command would abort.

Whenever an error is encountered during the execution of a macro, the macro aborts (unless the **ABORTOFF** command has been issued), and the TIMESLAB prompt is reissued. If the error can be easily fixed (for example by defining a variable that the macro needs), the macro can be restarted by issuing the command **MACRO** with no argument. For more information on this see Section B.7.

The error checking has been done to try to make sure first that TIMESLAB doesn't "bomb" (i.e., doesn't encounter an error that causes an immediate return to DOS, or what's worse, requires the system to be rebooted), and second that the user will be alerted to the existence of improper commands as much as possible.

B.4.7. On-line and User-Definable Help Facilities

Much of the content of this manual is available through the command **HELP**. Further, users can define their own help facilities via the **NOTES** command.

On-line Help

Issuing the command **HELP** results in a list of commands and a set of area names being placed on the screen. To get a brief description of what a command does and a list of its arguments (if it has any) one can issue the **HELP** command with the command name in parentheses. To get a list of commands that are available in one of the listed areas, the **HELP** command with the name of the area in parentheses can be issued.

Special help screens are available while within either the TIMESLAB editor or the graphics locator/editor **FIND**. The method of accessing these facilities is indicated on the screen whenever **FIND** or **EDIT** is being used. Also within **FIND**, one can get a list of the 256 ASCII characters and their ASCII codes. This same list is available from the TIMESLAB prompt by entering the command **HELP(ASCII)**.

The usual **HELP(topic)** command can also be used from within **EDIT** by first striking Alt-F1. Thus while using **EDIT** to create a macro file one can recall what the arguments are for a particular command.

User-Definable Help

A user can provide further help facilities by first forming a file called a notes file using **EDIT**, and then using the **NOTES** command to retrieve the notes that have been defined. This method can be used to keep track of the names and locations of macro or data files, or to store information about an analysis that has been performed. A notes file consists of a text file (creatable via the **TIMESLAB** editor or any other text editor that can create a pure ASCII file) containing a series of what **TIMESLAB** calls notes. A note consists of a first line containing a note name (as many as 15 characters starting in column one having no spaces) followed by as many lines (having as many as 72 characters each) containing the note. Each of these note lines must have a blank character in column one.

Once a notes file has been created, entering the **NOTES** command with the name of a file in parentheses results in the display of the list of note names in that file. Entering **NOTES(filename,notename)** results in a display of that note.

B.4.8. Creating and Saving Arrays

As we mentioned above, the basic objects of interest in **TIMESLAB** are arrays. There are two methods for creating arrays: (1) reading them from a disk file (via the **READ** command), and (2) as output from a command. Once an array has been defined, it can be saved to a disk file by the command **SAVE(X,n,fname)**, where **fname** is the name of the file to receive the array **X**. Unless an array is saved to disk in this way, it is lost when the current **TIMESLAB** session ends.

The user can form a file on disk containing an array using any text editor (including the **EDIT** command) that is capable of producing a pure ASCII file. The **TIMESLAB** editor is specially designed to form and modify the short data and macro files that **TIMESLAB** uses. Getting in and out of **EDIT** is done quickly and easily. If some other editor is used, care must be taken that the result contains no non-ASCII characters as the **READ** command will not recognize them. This includes for example the tab character used by many editors.

TIMESLAB Data Files

TIMESLAB will accept data files of two kinds: (1) formatted or (2) free formatted (the **SAVE** command saves an array to disk in formatted form). For either kind the first line of the file must be a label identifying the array. This label can have as many as 40 characters. Once **TIMESLAB** has read the file this label is used as a heading for lists and plots of this array.

In a formatted file, the second line of the file must contain the number of elements in the array followed by a Fortran **FORMAT** statement (including its

parentheses) describing the format that the elements of the array have been entered in the file. The array length and format can be anywhere on this second line as long as (1) the length comes first, (2) there is at least one space between it and the format statement, and (3) the format statement has no more than 30 characters. The format statement must include the beginning and ending parentheses.

In an unformatted file, the second line contains only the length of the array. After this second line the elements of the array are entered in any format desired as long as there is at least one space separating consecutive elements. In a formatted file, the elements of the array must follow the second line and be entered according to the format given on the second line. This ability to have formatted files allows the user to have identifying information in the file or to skip columns of numbers in the file if desired.

Note that the **SAVE** command makes it possible to append the array being saved onto the end of an existing file, and that the **READ** command allows one to skip several data sets on a file before reading the one that is desired.

To illustrate data files, we include a listing of the file **AIR.DAT** which consists of a formatted data file and is Series VI in Chapter 1. Using the formatted form for **AIR.DAT** allows one to put the year into the file. The format statement tells **TIMESLAB** to skip the year when reading the data.

Airline Passengers, 1949-60

144	(5X,12F5.0)											
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	606	508	461	390	432

There are two other points to be made about data files. First, several data sets can be placed in the same file. In this case, each one must be in the form described above. Then the four argument form of the **READ** command can be used to read them one at a time (see the **REG** macro for an example of how **TIMESLAB** can be tricked into reading a data file that has several data sets stored by column). Second, a data point that is represented in the data file by just a decimal point will be read as a zero. Thus one can represent missing values by a decimal point.

B.4.9. Overwriting Previously Defined Variables

When TIMESLAB is first started, it is in what is called the “overon” mode; that is, if the user enters a command that has as output a variable that has already been defined, TIMESLAB will not ask permission to overwrite the new value onto the old. One can issue the type 5 command **OVEROFF** to tell TIMESLAB to henceforth ask for permission to overwrite a variable any time that a command would result in such overwriting. The command **OVERON** puts TIMESLAB back into the overon mode. If a user would like to always be in the overoff mode, the **OVEROFF** command can be included in the **AUTOEXEC.MAC** macro file (see Section B.4.14).

B.4.10. Printed Output

TIMESLAB can produce printed output in several ways. See Section B.5 for a description of how to obtain hard-copy versions of graphics screens, and see the descriptions of the various printing commands in Sections C.1 and C.2.

B.4.11. The DOS Intraline Editing Functions

The DOS command line special intraline editing functions work when at the TIMESLAB prompt in the same way as at the DOS prompt. Thus when at the ? prompt (in the text mode only), pressing the F3 function key brings back the last command that was entered, pressing F1 repeatedly brings back one character at a time from the previous command, pressing F2 and a character *x* copies all characters from the previous command up to but not including *x*, and pressing F4 and a character *x* followed by F3 brings back the previous command with the characters up to *x* deleted. Characters can be inserted into the previous command by pressing F2 followed by the character before where the insertion is to be made, pressing the Ins key, typing the desired insertions, and then pressing F3 to bring back the rest of the line. Characters can be deleted in the previous command by a similar process except using the Del key to delete the desired characters. Using these special functions one can:

1. Instantly repeat a command by pressing F3 and return.
2. Edit and reenter a command without having to retype the whole line.
3. Form a new command that is similar to the previous command.

The DOS manual has more details about these editing functions.

B.4.12. The TIMESLAB DOS Mode

TIMESLAB has a special type 5 command called **DOS**. Entering this command results in the prompt `?d>` where `d` is the current default disk drive. At this point TIMESLAB is essentially in a "DOS mode," wherein any DOS command can be issued, including invoking any text editor or copying files. To exit this DOS mode, one answers the prompt with the word **exit** in either upper or lowercase. This returns control to the usual TIMESLAB prompt.

To use the **DOS** command, TIMESLAB must be able to access a copy of **COMMAND.COM** either by having one in the same (sub) directory as **TIMESLAB.EXE** or by having a **PATH** to it (see the DOS manual). If TIMESLAB can't find **COMMAND.COM** or the command issued at the DOS prompt would result in not having enough RAM, TIMESLAB will issue a message to that effect. For example, TIMESLAB uses approximately 480K of RAM and DOS uses anywhere from around 20K for Version 2.0 to around 60K for Version 3.2. Thus if one has Version 3.2 of DOS, only programs requiring less than around 100K RAM could be executed from the TIMESLAB DOS prompt ($480 + 60 + 100 = 640$).

B.4.13. The TIMESLAB Full Screen Text Editor

A full screen editor specially designed for the relatively short data and macro files used by TIMESLAB can be invoked by issuing the **EDIT** command (see Section B.6). This editor uses a subset of the features of the IBM Professional Editor.

B.4.14. AUTOEXEC.MAC

If there is a file called **AUTOEXEC.MAC** on the default disk drive when TIMESLAB is invoked, then TIMESLAB will execute it as a macro. This ability can be used in a similar way as **AUTOEXEC.BAT** in DOS.

B.4.15. Random Number Generator Seeds

Several of the TIMESLAB commands generate random numbers and thus require a "seed" (a real scalar used to start the random number generator). The user can either choose to specify a seed, in which case an experiment can be rerun at a later time with the same results if the same seed is used, or choose to use a seed that TIMESLAB will generate. To get this default seed, the user need only enter 0. as the argument in the command being used. Any time one of the simulation commands is issued, TIMESLAB generates an output seed that is stored internally. The default seed for a command is in fact this output seed from the last time one of the simulation commands was used, unless the present command is the first time a simulation command has been issued, in which case TIMESLAB will find a default seed by reading the internal clock

on the PC.

B.4.16. Keeping a Record of a Session

The command

```
RECORD(fname)
```

followed later by the command

```
RECORD(close)
```

will keep a record of the results displayed by several commands (see the description of the **RECORD** command in Section C.2). The **RECORD** command also keeps a record on the file of every command that is entered from the keyboard. This feature makes it possible to later create a macro by editing the file of commands.

If the filename is one of the valid names for the printer (for example, `lpt1` or `prn`), then the record is directed to the printer.

B.4.17. Exiting from TIMESLAB

To exit from TIMESLAB and return to DOS, one need only answer the prompt with the type 5 command **QUIT**. Any arrays that have not been saved onto a disk file either by the **SAVE** command or by using the TIMESLAB editor will of course not be available the next time TIMESLAB is used.

B.5. TIMESLAB Graphics

The TIMESLAB program has been designed with the graphical nature of time series analysis in mind. The commands that produce high resolution pixel graphs on the screen are:

- | | |
|---------------|---|
| HIST | - Simple histogram plot. |
| INFQNT | - Produce an informative quantile plot of a data set. |
| PLOT | - Simple Y versus X plot. |
| PLOT2 | - Superimpose the plot of two Y variables versus the same X variable. |
| PLOTK | - Superimpose the plot of as many Y variables as desired, each versus its own X variable. |
| PLOTSP | - Spectral density plot (log scale). |

- PLOTCSF** - Spectral density plot (nonlog scale).
- RESCREEN** - Reproduce a graph that was previously saved to a file using the **SAVESC** command.

In this section we give detailed information about how these commands and others can be used to provide interesting graphical displays for studying time series analysis.

B.5.1. Specifications for Plots

The graphics routines listed above all produce plots consisting of a horizontal and vertical axis (with numerically labeled tic marks on both axes), one or more graphs on the axes, and one or more captions above the plots. Because of the lack of resolution on the screen, there are no descriptive labels below the horizontal axis or to the left of the vertical axis. Such labels can be added to graphs by the user using the **FIND** utility (see below) or the two- or three-argument forms of the **LABEL** command (see the **HAT2** macro for an example of using the **LABEL** command).

The following values determining the size and location of plots can be specified by the user using the **LOTSIZE** command:

- npixx,npixy** - Length in pixels of the horizontal and vertical axes (defaults are 480 and 120).
- nx1,ny1** - Pixel location of the origin (lower left-hand corner of the screen is (0,0), defaults are 55 and 30).
- nticsx,nticsy** - Number of tic marks on the horizontal and vertical axes (defaults are 10 and 10).
- nclabx,ndecx** - Numerical labels at the tic marks on the horizontal axis have a total of **nlabx** characters (including the decimal point and any minus sign) with **ndecx** places to the right of the decimal point (defaults are 8 and 2).
- nclaby,ndecy** - Numerical labels at the tic marks on the vertical axis have a total of **nlabx** characters (including the decimal point and any minus sign) with **ndecy** places to the right of the decimal point (defaults are 6 and 2).

Note that each character in a tic mark label is made up from a pattern of 8×8 pixels being on or off. There is no space between the 8×8 pixel regions as the spacing is supplied by columns in the characters being off. The

bottoms of the tic mark labels on the horizontal axis are placed 12 pixels below the axis. In addition, one should reserve the bottom eight pixels of the screen for the “graphics menu” (see below). Thus **ny1** should be at least 20. All of the graphics commands leave a space of two pixels between the end of the vertical axis tic mark labels and the vertical axis. Thus **nx1** should be at least **8*nclabx+2**.

B.5.2. Graphics “Windows”

By default, TIMESLAB produces plots that fill much of the PC’s screen, and unless the **PLOTON** command has been issued, any plot that is produced is erased before another can be produced. It is possible (by using the **PLOTON** and **PLOTSIZE** commands) to divide the screen into different areas and produce graphs in more than one of them at the same time. These areas are called windows although they are not strictly windows in that commands can produce graphics that intrude upon more than one area of the screen. The following strategy can be used to do windowing:

1. The **PLOTON** command is used to tell TIMESLAB not to switch back to the text mode between commands. The prompt will now be reissued at the top of the screen after each command so that it will not overwrite any graphs already on the screen. Any command that results in displayed output may overwrite graphs that are on the screen. There is no separate window for displaying nongraphical material.
2. The **PLOTSIZE** command can be used to specify the size and location of subsequently produced plots. Users can make these specifications anything that they want, but issuing the command **PLOTSIZE(k)** for the values of **k** in the table below sets up certain standard windows. The values in the table have been designed so that different regions of the screen will not interfere with each other. For example, regions 1 through 4 do not overlap unless the user has increased **nclabx** or **nclaby** from the default values.

This strategy allows two important uses of graphics: (1) several graphs can be established on the same screen (see the **COS** macro for an example), and (2) many graphs can be superimposed on the same axes (see the **HAT2** macro for an example). The **BATCHON** command is important in this strategy. Ordinarily, the user wants the graphics menu to appear after producing a plot so that it can be printed or saved to a file. Unfortunately, this means that the user must strike a key in between each graph. If the **BATCHON** command is issued, then there is no pause between plots. Thus before the last plot in a series of plots is formed, one might want to issue **BATCHOFF** so that the graphics menu will appear after all of the plots are on the screen. One can also use the **GRMENU** command to place the graphics menu on the screen at any time that the screen is in the graphics mode.

Standard Windows Using the PLOTSIZE(k) Command

k	Region of Screen	npixx	npixy	nx1	ny1
0	Full screen	480	120	55	20
1	Upper left	260	60	50	120
2	Lower left	260	60	50	20
3	Upper right	260	60	360	120
4	Lower right	260	60	50	20
5	Left half	260	120	50	20
6	Right half	260	120	360	20
7	Top half	480	60	50	110
8	Bottom half	480	60	50	20

B.5.3. The Graphics Menu

Whenever TIMESLAB produces a graph on the screen (unless the **BATCHON** command has been issued), it places a message (called the graphics menu) at the bottom of the screen (in the bottom eight rows of pixels) alerting users to the fact that they can perform several useful operations, and then pauses until users either perform one of these operations or indicate that they want to go on to the next command. It is also possible to place this graphics menu on the screen at any time the screen is in the graphics mode by issuing the **GRMENU** command.

The operations that are available from the graphics menu are determined by striking one of the following function keys (striking any other key will result in the TIMESLAB prompt being reissued):

- F1** - This invokes the **FIND** utility; that is, an interactive locating and labeling facility becomes available (see Section B.5.5).
- F3** - This causes whatever is currently on the screen (with the graphics menu erased) to be printed on whatever printer is attached to the PC. In order for this to operate properly, the **PRINTSEL** command has to have been issued previously to tell TIMESLAB what type of printer has been attached. See Section B.5.4 for more details about obtaining such screen dumps.
- F5** - This causes TIMESLAB to create a disk file (it will ask the user for the name of the file to be formed) containing a com-

pressed version of what is currently on the screen, again with the graphics menu erased. This compressed screen image can be instantly redisplayed later using the **RESCREEN** command.

- F9** - This allows the user to break out of a macro if one is being executed.

If the **BATCHON** command has been issued, then the graphics menu is not displayed and **TIMESLAB** does not pause to allow the user to see the completed graph. If the **SAVEESC** and/or **PSON** commands have been issued, then the completed graph will be saved to a file or dumped to the printer before the next command is executed. See the **COS** macro for an example of how selectively issuing the **BATCHON** and **BATCHOFF** commands can be used to obtain several plots on the screen with no pause until after the last one is formed.

B.5.4. Graphics Screen Dumps

Any time that a graph is on the screen, a pixel-by-pixel image of the screen can be produced on the printer that is attached to the PC as long as two requirements are met:

1. The printer is capable of performing high resolution graphics. Examples of this are: (a) the IBM Graphics Printer or any printer that claims to be compatible with it (these are 9 pin dot matrix printers), (b) a Toshiba 321 24 pin dot matrix printer or any printer that claims to be compatible with it, and (c) the Hewlett-Packard Laserjet Printer or any laser printer that is compatible with it.
2. The PC has been alerted to the fact that screen dumps will be requested.

The manner in which the second requirement is met depends on whether or not the printer attached to the PC is one of the types for which **TIMESLAB** has a built-in screen dump capability. These types are the three that are listed in item 1 above. If the attached printer is one of these three types, then one need only use the **PRINTSEL** command to tell **TIMESLAB** which type it is, and then graphics screen dumps can be obtained in either of the following two ways:

1. By issuing the **PSON** command prior to producing the graph. This method will produce a screen dump for any graphics screen that is produced until the command **PSOFF** is issued and is included in **TIMESLAB** primarily for use within macros.
2. By striking **F3** from the graphics menu.

If the PC is attached to a laser printer, there are a few other considerations involved in producing screen dumps. First, communication between the printer and the PC must have been established. The manual for the printer should contain information about how this is done. Second, the image constructed on the printer is approximately 4.25 inches wide and 2.5 inches high. Thus only three such images will fit on one page. After three images are produced, issuing the command **PAGE** will cause the page to be ejected. The third consideration with laser printers is how much "on-board" memory the printer has. For example, the H-P Laserjet Printer has only enough memory to produce one image on a page, while the Laserjet Plus can produce more than the three which will fit.

Screen Dumps on a Nonstandard Printer

If the printer that is attached to the PC is not one of the three types described above (this can only really be determined by trying the methods described above), then the user will have to rely on the ability of DOS itself to produce graphics screen dumps (on the standard printers, TIMESLAB does the screen dumps). In this case, the screen dump is produced by striking the Shift and PrtSc keys at the same time while the graph is on the screen. When this is done, DOS takes control from TIMESLAB and tries to do a screen dump. Unfortunately, screen dump software is not a standard part of DOS. Thus the user must have access to such software for their particular combination of PC and printer. For example, issuing the DOS command **GRAPHICS** prior to invoking TIMESLAB will make it possible to do screen dumps to the IBM Graphics Printer via the Shift-PrtSc method. In general, the user must have software analogous to **GRAPHICS.COM** for their PC-printer combination, and have issued it prior to entering TIMESLAB.

B.5.5. The **FIND** Utility

Striking **F1** from the graphics menu invokes what is called the **FIND** utility. This utility sets up a graphics cursor on the screen that can be moved to any position on the screen by using the keys on the numeric keypad. The *x* and *y* values corresponding to the pixel at the lower left-hand corner of the cursor are displayed in the upper right-hand corner of the screen.

Locating Points on a Graphics Screen

When **FIND** is invoked, striking any of the keys labeled 1, 2, 3, 4, 6, 7, 8, 9 on the numeric keypad moves the cursor 10 pixels in the direction indicated by the key. For example, striking the key labeled 7 on the numeric keypad causes the graphics cursor to move up 10 pixels and 10 pixels to the left. The distance traveled by the cursor can be changed to any one of 1 through 10 pixels by

striking the corresponding function key. As the cursor moves, the x and y values are updated. Large moves of the cursor are also available by striking combinations of the Ctrl key and numeric keypad keys. For example, Ctrl-right arrow moves the cursor all the way to the right, Ctrl-PgDn moves it all the way to the bottom, and so on. The cursor can be moved directly to a specified (x, y) coordinate by striking Ctrl-F1. Note that all possible key strokes are described on the **FIND** help screen that is available by striking Alt-F1 while in **FIND**.

Labeling Graphs

While in **FIND**, the user can place any of the 256 ASCII characters anywhere on the screen. Some of these characters can be entered by just striking their key on the keyboard. To enter a character that doesn't have a key, one enters its ASCII code while holding down the Alt key. When the Alt key is then released, the character will appear. A list of the ASCII codes for the 256 available characters is available within **FIND** by first going to the **FIND** help screen (by striking Alt-F1) and then striking F1. One then returns immediately to the screen being edited by striking any key. Messages guiding the user through this process are placed on the screens involved.

Other Features of FIND

One can easily draw lines and rectangles on the screen by using combinations of the Alt key and function keys. The **FIND** help screen contains instructions on how to do this. Striking Ctrl-F9 while in **FIND** results in a blank screen that can be used to construct screens (from the characters and rectangles) that can later be printed or stored. Also while in **FIND**, Alt-F2 is a toggle for turning on and off the display of the current (x, y) values, and Alt-F3 is a toggle for turning the cursor "on and off." Actually Alt-F3 toggles whether the cursor is in the foreground or background color. This is important when placing the cursor in a solid rectangle wherein it would be invisible if one didn't have the ability to make it the background color.

There is one kind of graphics screen where **FIND** operates a little differently. There is a **TIMESLAB** command called **RESCREEN** which places a screen image (that has been previously saved onto a file by either striking F5 at the graphics menu or using the **SAVESC** command) onto the screen. When **FIND** is invoked after **RESCREEN**, **TIMESLAB** doesn't have any way of knowing the real-world environment of the screen. Thus there is no display of (x, y) as the cursor moves.

Exiting from FIND

To exit from **FIND**, one strikes Alt-F10. This puts users back into a situation where they can either print the screen, save it for later retrieval, or reenter **FIND**. At this point, striking any key other than F1, F3, or F5 gets back to the **TIMESLAB** prompt.

B.5.6. Saving and Retrieving Graphics Screen Images

Some very effective graphics screens (and thus hard copies of screens) can be constructed by using the ability of **TIMESLAB** to store and retrieve files containing compressed graphics screens. A typical graph produced by **TIMESLAB** can be stored into a file of approximately 3000 to 4000 bytes. Thus 90 or 100 such screens can fit on one floppy. There are two ways to save a graphics screen to a file:

1. By striking F5 at the graphics menu. **TIMESLAB** then asks for the name of the file to receive the compressed screen image.
2. By issuing the **TIMESLAB** command **SAVESC** prior to issuing the command that places the graph on the screen. **SAVESC** has one argument, namely the name of the file to receive the image.

Once such files are formed, one can then "rescreen" one or more of them by issuing the **RESCREEN** command from the **TIMESLAB** prompt. To rescreen one image, the command is **RESCREEN(fname)** where **fname** is the name of the file containing the image. To rescreen several images, the **RESCREEN** command can be issued with the names of the files containing the screens (in the desired order) as arguments, together with a final argument that is 1 if each screen is to be erased before the next is placed on the screen, or 2 if the screens are to be superimposed. If many images have been stored on files having names that all begin with the same set of characters and end with consecutive integers (e.g., **wn3,wn4,wn5,wn6**), then they can all be rescreened by issuing a command of the form **RESCREEN(wn,3,6,iopt)**, where **iopt=3** means erase in between and **iopt=4** means don't erase in between. Thus one can superimpose images or else just see a series of them one after another. This happens fast enough (the speed depends on the speed of disk accesses on the machine being used) that one can create animation type effects. If the images are stored on a floppy, there is perhaps a one-second pause between images while on a hard disk there is about half this pause. If the screen images are stored on a RAM disk, there is hardly any pause between rescreened images. Note that a macro can be used to create the files containing the images.

Once the final image from a **RESCREEN** appears on the screen, the graphics menu is placed on the screen so that this (possibly new) image can be edited using **FIND** and then printed or saved.

B.5.7. Scaling and Labeling Graphs

Some very important parts of the appearance of any of the graphs produced by TIMESLAB are the caption above the graph, the graph's scale, and the numerical labels placed at the tic marks on the graphs.

Caption and Axis Labels

TIMESLAB places the label (or labels if appropriate) associated with the array (or arrays) being plotted above the graph. Because of the lack of resolution on the screen, there is no textual label placed to the left of the vertical axis or below the horizontal axis. The FIND utility can be used to add such labels if desired, or to place text anywhere on the screen. There are also two forms of the LABEL command which can be used to add labels to plots.

Scaling

Each of the PLOT, PLOT2, PLOTK, HIST, and PLOTCSF commands can be given in two forms. The first form (with no scaling information) tells TIMESLAB to use the actual minimum and maximum values of what is being plotted as the values at the ends of the axes. The second form (with scaling information) tells TIMESLAB to use the minimum and maximum values entered as arguments as the values at the ends of the axes. If the second form is used and there is a value to be plotted that is outside of the user-specified range, TIMESLAB reverts to the first form. Typically then, one first uses the first form (unless there are some natural limits on the graphs such as -1 to 1 for a correlogram), inspects the resulting plot, and then reissues the command in the second form with some esthetically appealing limits. Being able to specify limits like this also allows several plots to be produced with the same scale. To illustrate these ideas consider Figure B.1 which consists of hard copies of the graphs produced by the commands

```
x=WN(12345,100)
LABEL(x)='PLOT(x,100)'
PLOT(x,100)
LABEL(x)='PLOT(x,100,0,100,-4,4)'
PLOT(x,100,0,100,-4,4)
```

In this case we could anticipate what scaling information would lead to a pleasing graph.

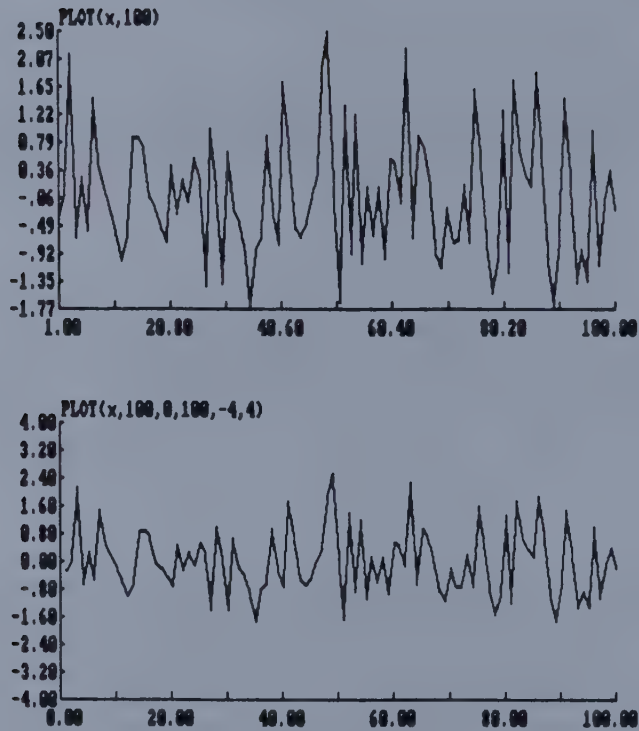


Figure B.1. The Effect of Using Scaling Information in the PLOT Command.

Tic Mark Labels

For the vertical axis in PLOTCSF and for both axes in PLOT, PLOT2, PLOTK, and HIST, the user can determine what the tic mark labels are. TIMESLAB attempts to put numerical labels at as many of these tic marks as possible. It will also always put labels at the intersection of the two axes. If a numerical label is too large to fit the format specified by `nclabx`, `ndecx`, `nclaby`, and `ndecy`, then a string of asterisks is placed at the tic mark instead. If this happens, the user must rescale the array or arrays being plotted so that they will fit. This can be annoying but rarely happens and it is easy to do the rescaling.

To illustrate how appealing labels at the tic marks can be obtained, consider how the horizontal axis is labeled. The minimum value (either the actual minimum for the first form or the inputted value for the second form) is placed at the intersection of axes. The label at the first tic mark would then be the minimum value plus a fraction of the range of the axis (the maximum value

minus the minimum value) equal to the number of tic marks on that axis. TIMESLAB tries to place this label at the tic mark with the decimal point directly under the mark. If the label would not fit without interfering with the one already positioned, TIMESLAB goes on to the next tic mark and repeats the process except that the label would be the minimum value plus a fraction of the range equal to two over the number of tic marks. The vertical axis is treated in exactly the same way.

As an example of producing nice tic mark labels, suppose the array being plotted on the horizontal axis consists of the numbers 1, 2, \dots , n . In this case, the user most likely will want the tic mark labels to be whole numbers. Using zero and the next multiple of ten greater than or equal to n as the endpoints of the axis will give such labels if one is using ten tic marks. Note that such a strategy can be used to move a graph away from the axes as well.

B.6. The TIMESLAB Text Editor

During a TIMESLAB session one often needs to create or modify macro or data files. The EDIT command allows this to be done without going to the trouble of leaving TIMESLAB or going to its DOS mode in order to use an external text editor. EDIT is a simple full screen editor; that is, a portion of a file is placed on the screen and changes can be made directly on the screen. Thus the user positions the cursor to where changes are to be made and then enters the changes directly by either replacing a character, inserting a character or a line, or deleting a character or a line.

EDIT is not intended to have all of the features of a full screen editor such as the IBM Professional Editor. It has no block copies, moves, or deletions, or the ability to make global searches or replacements, or the ability to merge text from another file. Rather it is intended to be very simple to use to form and modify the kinds of files that TIMESLAB uses (particularly data and macro files). Files having as many as 2000 lines and 18,000 characters can be edited using EDIT.

Entering and Exiting the Editor

To edit an existing ASCII file, one need only enter the command

```
EDIT(fnold,fnnew),
```

where **fnold** and **fnnew** are the names of the existing file and the file to receive the modified version of the file. If the user wants these names to be the same, then the command can be issued as **EDIT(fname)**. If there is no file on the default drive having this name, then EDIT will assume that a new file is to be created.

To leave the editor and return to the TIMESLAB prompt, one has two choices: F3 which saves the file and returns to the TIMESLAB prompt, and

F4 which returns directly to the TIMESLAB prompt without saving the new file. Usually one uses F3 but sometimes it is helpful to start over with a fresh copy of the old file. Then one can use F4 and call EDIT again. In either case, EDIT will ask for verification that the user genuinely wants to leave EDIT.

Adding and Deleting Lines

After entering the EDIT command the user sees an introductory screen that indicates whether an existing file is to be edited or a new file is to be created. If a new file is being created, lines marking the beginning and end of text will appear on the screen with the cursor at the first line. To begin entering text, one strikes F5 to get into the "insert lines mode". Striking a carriage return at the end of a line causes EDIT to give a new blank line to use to enter the next line. When all lines have been entered, the only way to exit this insert lines mode is to strike another carriage return while a new line is still blank.

If an existing file is being edited, the initial lines of that file will be placed on the screen and the cursor will be at the top line of the screen. Then the cursor can be moved to where changes are to be made, and the changes made. At any time in an editing session, one can insert lines by positioning the cursor to the line before where the lines are to be inserted, and then striking F5. This places EDIT into the insert lines mode, and again the only way to leave this mode is to strike a carriage return on a newly opened blank line.

To delete a line, one need only position the cursor to the line to be deleted, and then strike F6.

Positioning the Cursor

One moves the cursor by using the keys on the numeric keypad together with the Del, Ins, Tab, and Ctrl keys. The keys on the numeric keypad will only work properly if the keyboard is in the Num Lock off state. If a number appears on the screen when one of the keys on the numeric keypad is struck, then the keyboard is in the Num Lock on state. If this happens, one need only strike the Num Lock key to reverse the state.

The following keystrokes move the cursor to the indicated positions:

- | | |
|---------------|---|
| Arrows | - One position in the direction of the arrow. |
| Ctrl→ | - Seven positions to the right. |
| Tab | - Seven positions to the right. |
| Ctrl← | - Seven positions to the left. |
| F9 | - To the left or right end of the current line. |

Home	- To the top of the screen.
End	- To the bottom of the screen.
Pg Up	- Up 12 lines.
Pg Dn	- Down 12 lines.
Ctrl-Home	- To the beginning of the file.
Ctrl-End	- To the end of the file.

Deleting, Overwriting, and Inserting Characters

To delete the character where the cursor is located, one need only strike the Del key. To delete the character to the left of the cursor, one strikes the backspace key.

When the cursor is located at a certain character on the screen, two things can happen when one of the usual alphanumeric keys is struck: either the current character is replaced by the struck character or the struck character is inserted before the character where the cursor is. These two modes are called the overwrite and insert modes and which will happen is controlled by the Ins key. When EDIT begins it is in the overwrite mode. The Ins key is used to switch back and forth between the two modes. One can always tell which mode is in effect because the cursor is normal size in the overwrite mode and larger than normal when in the insert mode.

HELP Within EDIT

While in EDIT one can get a screen that gives directions on how to use the editor. This is done by striking F1. By striking Alt-F1, one can also get the usual TIMESLAB HELP. This is particularly useful when forming a macro file. Thus if users forget the arguments of some command that they want to use in a macro, they can strike Alt-F1, at which point TIMESLAB asks which help topic users want information about. Once the information is on the screen, one returns to editing (right where they were) by striking any key.

B.7. Using Macros

The ability to execute macros makes TIMESLAB an interpreted programming language. The purpose of this section is to discuss in detail the features of TIMESLAB macros. In the next section we discuss several examples.

B.7.1. Basic Features

A macro is a collection of TIMESLAB commands residing on a disk file. The user invokes the macro by entering the command **MACRO(fname)** where **fname** is the name of the file containing the commands to be executed.

Example

To illustrate some of the features of macros, we include the example below. This example is contained in the file **EXAMPLE.MAC** and we will refer to it as the **EXAMPLE** macro. Note that we have inserted line numbers in this and all other macros in the book. These line numbers are not and should not be included in the actual file. **EXAMPLE** does very few calculations and is only intended to illustrate the basic form and structure of macros. Note that this example contains a large number of comments, that is, lines that begin with one or two semicolons and lines that contain a semicolon followed by a comment.

```

1 ;;
2 ;;   EXAMPLE.MAC: This macro illustrates the basic features of
3 ;;                   macros. All that it does is find the sum of the
4 ;;                   first n natural numbers for a user-specified value
5 ;;                   of n. The value of n must be less than or equal
6 ;;                   to 10. If you haven't defined n, break the
7 ;;                   macro now and define it. Then re-invoke the
8 ;;                   macro by issuing the command MACRO with no
9 ;;                   argument.
10 ;;
11 ;;   INPUT: n (an integer between 1 and 10, inclusive)
12 ;;
13 ;;   OUTPUT: sum (an integer containing the sum)
14 ;;
15 PAUSE ;this lets user break and define n if necessary
16 ;
17 ;   by having the next line, the prolog of the macro can be skipped
18 ;   by entering the command MACRO(example,start)
19 ;
20 ;start
21 IF(n.lt.1,error) ;check if any error in value of n
22 IF(n.gt.10,error) ;if yes, go to line labeled error
23 GOTO(st1) ;if no, go to line labeled st1
24 ;
25 ;error
26 ;
27 PROMPTON ;turn prompt back on so that error message will be seen
28 ;;
29 ;;   n must be between 1 and 10, inclusive
30 ;;
31 GOTO(end) ;now exit the macro because of error
32 ;
33 ;st1

```

```
34 ;
35 ;;
36 ;;   Now we will find the sum by five methods. I'll
37 ;;   pause after each one.
38 ;;
39 PAUSE
40 ;method1
41 CLS   ;clear screen
42 ;;
43 ;;   Implied loop using IF and GOTO:
44 ;;   This is moderately fast.
45 ;;
46 i=1
47 sum=0
48 ;s1
49 sum=sum+i
50 IF(i.eq.n,e1)
51 i=i+1
52 GOTO(s1)
53 ;e1
54 LIST(sum)
55 PAUSE
56 ;method2
57 CLS   ;clear screen
58 ;;
59 ;;   Implied loop using three-argument IF:
60 ;;   This is the fastest of the four loops.
61 ;;
62 i=0
63 sum=0
64 i=i+1
65 sum=sum+i
66 nmi=n-i
67 IF(nmi,-3,1,-3) ;nmi=0 means stop loop
68 LIST(sum)
69 PAUSE
70 ;method3
71 CLS   ;clear screen
72 ;;
73 ;;   IF...ENDIF construct:
74 ;;
75 i=1
76 sum=0
77 ;s2
78 IF(i.le.n)
79     sum=sum+i
80     i=i+1
81     GOTO(s2)
82 ENDIF
83 LIST(sum)
84 PAUSE
85 ;method4
86 CLS
87 ;;
```

```

88 ;;   WHILE-END Construct:
89 ;;   This is slow because it must back up in the file.
90 ;;
91 i=1
92 sum=0
93 WHILE(i.le.n)
94 sum=sum+i
95 i=i+1
96 END
97 LIST(sum)
98 PAUSE
99 ;method5
100 CLS ;clear screen
101 ;;
102 ;;   Fast Way:
103 ;;
104 nn=LINE(n,0,1) ;form array 1,...,n
105 one=LINE(n,1,0) ;form array 1,...,1
106 xsum=DOT(nn,one,n) ;find inner product of nn and one
107 sum=xsum ;convert xsum to integer
108 LIST(sum)
109 PAUSE
110 ;
111 ;end
112 ;
113 ;;
114 ;;   This is the End of the EXAMPLE macro
115 ;;

```

Commenting TIMESLAB Code

As in any programming language, it is important to put comments into a macro file. There are two kinds of comments that can be used: (1) nondisplayed comments and (2) displayed comments. The first begin with a single semicolon and the second with two semicolons. A nondisplayed comment is ignored by TIMESLAB, while a displayed comment is not executed but is displayed on the screen (unless the **PROMPTOFF** command has been issued). Note that a nondisplayed comment line can be used to provide a target for the **IF** and **GOTO** commands (see below). If a semicolon appears in a line of code, then it and anything to its right on the line are ignored by TIMESLAB.

A displayed comment is any line that begins with two semicolons. Such a line can be used to document a macro, but also has another important function. When TIMESLAB encounters a displayed comment line in a macro, it displays it on the screen (unless the **PROMPTOFF** command has been issued). This allows a macro to tell the user what it is doing as it is being executed. Note that almost every macro in this book begins with a prolog consisting of displayed comments that tell the user what the macro does and what input is required.

Multiple Entry Points

A user can begin execution of a macro at any line in the file that begins with a semicolon and then has a string of characters by issuing the command

```
MACRO(fname,string)
```

where **fname** is the name of the file containing the macro, and **string** is the string of characters on the line where the macro is to start. In the **EXAMPLE** macro, we have provided several entry points, including the lines marked **start**, **method1**, **method2**, **method3**, and **method4**.

Interrupting a Macro and Providing Input

While a macro is being executed, the user can interrupt it by striking the F9 function key. This causes TIMESLAB to stop executing commands from the macro file and to start accepting commands from the keyboard again. To restart the most recently interrupted macro, one need only issue the command **MACRO** with no argument. Thus for example, one can start a simulation study, interrupt it and look at the results so far, and then restart it.

The ability to interrupt a macro can also be used to allow it to ask the user for input. The TIMESLAB command **PAUSE** causes TIMESLAB to pause in its execution of a macro until either F9 is struck (in which case the macro is broken and TIMESLAB accepts commands from the keyboard) or any other key is struck (in which case the macro resumes execution). Thus a macro can include some displayed comments telling the user to define some quantities and then a **PAUSE** command. When the macro is executing the user can break the macro, define the requested variables, and then restart the macro. Note how this idea was used at the beginning of the **EXAMPLE** macro.

Nesting Macros

One macro can call another macro. However, there can only be a total of five macros whose execution has not finished at any one time. Thus recursion (that is, a macro calling itself either directly or indirectly) is not allowed. Note that TIMESLAB has no provision for the passing of arguments when one macro calls another; for example, if a macro operates on an array called **X**, then the calling macro must have defined **X**.

Error Handling During Macros

When TIMESLAB starts, it is in the aborton mode; that is, an error in a macro causes the execution of the macro to stop at the line where the error occurred. If the error is easily corrected, for example, if a variable needed in the macro was not defined, then the user can correct it and restart the macro at the

line where the error occurred by issuing the command **MACRO**. If, however, it is necessary to modify the macro file in order to fix the error, then the command **MACRO(fname)**, where **fname** is the corrected version, must be given even if the new file has the same name as the old file. There is a command called **ABORTOFF** which tells **TIMESLAB** to continue executing the macro even when confronted with an error. The command **ABORTON** cancels the effect of **ABORTOFF**.

Other Basic Features

Other basic features of macros include:

1. When **TIMESLAB** starts, each line that is executed in a macro is displayed on the screen. This can be time consuming in a long macro. The command **PROMPTOFF** is provided to tell **TIMESLAB** to stop the displaying. The effect of this command is cancelled by the command **PROMPTON**.
2. Issuing the command **SPEAKERON** will result in having the speaker beep as each line in a macro is executed. The effect of this is cancelled by the command **SPEAKEROFF**.
3. Occasionally it is useful to execute a macro so that **TIMESLAB** will wait for a key to be struck before each command is executed. The command **SINGLEON** is provided to put **TIMESLAB** into this "single step" mode. The effect of **SINGLEON** is cancelled by the command **SINGLEOFF**.

B.7.2. Macro Branching Commands

There are five kinds of branching constructs available for use in macros. These constructs allow **TIMESLAB** to be used as a programming language. Each is illustrated in the **EXAMPLE** macro.

IF(var,no,zo,po)

This command compares the (real or integer) variable or immediate value **var** to zero and passes transfer to the line having offset **no**, **zo**, or **po** from the current line depending on whether **var** is negative, zero, or positive. Lines below the current line have positive offsets while those above it have negative offsets. In determining these offsets comment lines are not counted. This construct operates very quickly.

IF(exp,label)

In this command, the argument **exp** is one of **var1.lt.var2**, **var1.le.var2**, **var1.eq.var2**, **var1.ne.var2**, **var1.ge.var2**, or **var1.gt.var2**, where **var1** and **var2** are again real or integer variables or immediate values. If the ex-

pression is true then the next line to be executed is determined by the second argument as follows. TIMESLAB rewinds the MACRO file and looks for a line that starts with a single semicolon in column 1 followed by the same character string as is used as the second argument. If there is no such line, a message is displayed and the macro aborts.

```
IF(exp)...ENDIF
```

The one-argument form of the IF command, together with the ENDIF command, affords a traditional IF...ENDIF construct. The argument of the IF command is the same as the two-argument form. If the expression is true, then control passes to the next line of the macro. If it is false, control passes to the line following an ENDIF that matches the IF command (such IF...ENDIF forms can be nested).

```
GOTO(label)
```

This command provides an unconditional branch to the line determined by the argument in the same way as in the two-argument IF command above. Thus one can use implied loops.

```
WHILE(exp)...END
```

This pair of commands provides a while-end construct. Unfortunately this construct is not particularly fast, especially when the END is far from the WHILE.

B.8. Examples of Writing Macros

In this section we consider several examples of writing general purpose macros.

Example B.1	FORMING TABLES
--------------------	-----------------------

There is no command in TIMESLAB that is specifically designed to list a table on the screen. There are several ways that tables can be formed however. The simplest case is when we have several arrays that are all of the same length and we would like to list them in a table so that the first array forms the first column, the second forms the second column, and so on. The commands

```
x1=LINE(10,0,1)
x2=x1*x1
x=<x1,x2>
LISTM(x,10,2)
```

will form a table having two columns with the first being the numbers 1 through

10, and the second being the squares of the first. The **LISTM** command does not allow the user to specify the format of each row of the matrix being printed, and if there are more than six columns, those past the sixth in each row will wrap-around on the screen. In order to specify a format for a table, the **LIST** command must be used. The general form is

```
LIST(x,n,m,form)
```

where **x** is an array of length at least **n** whose first **n** elements are to be displayed on the screen with the first **m** in the first row, the next **m** in the next row, and so on, and the format of each row is determined by the argument **form**. Thus for the **x** of the example above, if we used

```
LIST(x,20,2,2f10.4)
```

we would not get what we want (the first row would be the first two elements which are 1 and 2, the next row would be 3 and 4, and so on). However, if we think of **x** as a (10×2) matrix, we can obtain an array for which the above command would work if we just transpose **x** first. Thus we could use

```
y=TRANS(x,10,2)
```

```
LIST(y,20,2,2f10.4)
```

In the rest of this example we consider a more complicated example of forming a table. The **DIST** command will calculate values of the pdf, cdf, or quantile function for the standard normal, Student's-*t*, χ^2 , or *F* distribution. The macro **NO1TAB** calculates the cdf for the standard normal for values of *z* between 0 and 2.99 in steps of .01. Then the ability of **TIMESLAB** to do concatenation is used to create a table of these values that has the values of *z* across the top and down the left side. Note that when the **LIST** command is used, the consecutive values of an array are listed across the rows of the list. Thus we need to create an array that starts with a zero, then the first ten cdf values, then the number .1 followed by the second ten cdf values, and so on. This is done by thinking of the cdf array as a matrix and using the **TRANS** command and the concatenation command (the type 4 command). Once an array containing the rows of the table is formed, we insert at its beginning the values that go across the top of the table. The resulting table is displayed in Table B.1. This table was actually created by using the **RECORD** command to place the table created by the macro onto a file which was then modified using the **EDIT** command. The resulting file was then inserted into the file which printed this book.

```

1 ;;
2 ;;   NOITAB.MAC: macro to calculate  $N(0,1)$  cdf at
3 ;;            $z=0,.01,\dots,2.99$  and write out table.
4 ;;
5 ;;           The RECORD command can be used to put this
6 ;;           table out to a file which can be edited to
7 ;;           get a nice table.
8 ;;
9 ;;
10 PAUSE
11 ;start
12 x=LINE(300,0,2.99,1)      ;Find values at which to find cdf
13 z=DIST(z,2,300,x)         ;Calculate cdf
14 z=TRANS(z,10,30)         ;Now append left margin of table
15 z1=LINE(30,-.1,.1)
16 z=<z1,z>
17 zz=TRANS(z,30,11)
18 z1=LINE(10,-.01,.01)
19 z1=<0,z1>
20 zz=<z1,zz>
21 form='f4.1,11f6.4'       ;Format for each row of the table
22 LIST(zz,341,11,form)    ;Write out table

```

Example B.2 A SIMULATION STUDY

In this example we construct Figure B.2 where we have superimposed the density estimators of a set of 1000 sample means and medians for $N(0,1)$ white noise series of length 10. Note how both graphs appear normally distributed and that the one for the sample means is more highly concentrated. This is interesting because asymptotic theory says that the limiting ratio of the variance of the sample mean to the variance of the sample median is $2/\pi$ as $n \rightarrow \infty$, which we are seeing here (the ratio of the values of normal pdf's at zero is the ratio of their standard deviations) for a sample size as small as 10. Note that Figure B.2 was produced by **MEDMN.MAC** with **nsamps=1000**, **n=10**, **dist=1**, **npts=50**, **rbins=4.**, and **kernel=3**. Since it is difficult to predict what a good scale would be on the axes in **PLOTK**, we waited to see the result of the macro and then used

```
PLOTK(y,fy,npts,2,type,-1.5,1.5,0,1.2)
```

to get the nice tic mark labels in Figure B.2, and then used the **FIND** utility (see Section B.5.5) to put the labels on the graph before getting a hard copy of the screen by striking **F3** from the graphics menu.

Note that this macro can be thought of as a model for almost any simulation study. It takes a long time to generate the 1000 samples since we are using **TIMESLAB** as an interpretive language. Once we have the arrays **xmn**

Table B.1. Table of Standard Normal Cumulative Probabilities

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
.7	.7580	.7611	.7642	.7673	.7703	.7734	.7764	.7794	.7823	.7852
.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986

and `xmed`, we can modify `npts`, `rbins`, and `kernel` and recall the macro by the command

```
MACRO(MEDMN,density)
```

which will go directly to the comment labeled `;density`, and thus avoid recalculating the means and medians.

This macro illustrates several other things as well. First, the `LINE` command is used in two different ways: (1) to form arrays `xmed` and `xmn`, each of length `nsamps` which will receive the means and medians for each sample, and (2) to form an array of length `n` which is used later to find the sample mean as an inner product of the sample and the array of ones divided by `n`. Next, the `PROMPTOFF` command is used at the beginning since echoing the thousands of commands to the screen will cause the macro to take much longer. After each

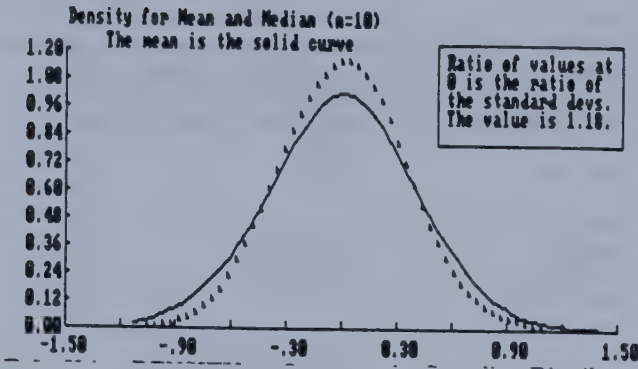


Figure B.2. Using DENSITY to Compare the Sampling Distribution of the Sample Mean and Median for Samples of Size 10 from a $N(0,1)$ Population.

mean and median is calculated, their value and which sample they correspond to are listed via the LIST command. Otherwise, the user would have no idea that anything was happening while the macro is executing. Third, note that the first time WN is called the user-supplied seed is used while henceforth using seed=0 causes WN to use the output seed of the last time WN was called.

Several times in the macro, individual elements of arrays are referred to by using the square brackets [...], while the ability to concatenate arrays using <...> is used a few times. Note that we found the median of a data set by sorting it and then finding the middle value (or the average of the two middle values if n is even). Finally, note the use of the implied loop using the IF and GOTO commands.

```

1 ;;
2 ;;   MEDWN.MAC: macro to generate nsamps white noise series
3 ;;           of length n (using seed as random number
4 ;;           generator seed) having distribution specified by
5 ;;           the integer dist and find the sample mean and
6 ;;           median for each one. Then the density command is
7 ;;           used to estimate the pdf for the means and medians.
8 ;;
9 ;;   INPUT: nsamps,n,seed,dist,  rbins,npts,kernel
10 ;;
11 PAUSE
12 ;start
13 ;
14 PROMPTOFF                ;don't waste time showing commands
15 xmn=LINE(nsamps,0,0)      ;reserve space for means
16 xmed=LINE(nsamps,0,0)    ;and medians
17 one=LINE(n,1,0)          ;we'll use this array in finding means

```

```

18 n1={n+1}/2           ;the median will be the average
19 n2={n+2}/2           ;of the sorted x's with these indices
20 on:=n
21 ns=1
22 x=W( seed,n,dist)     ;warm up random number generator
23 ;
24 ;startloop
25 ;
26 x=W(0,n,dist)         ;generate sample
27 x=SORT(x,n)           ;sort data
28 xmed[ns]={x[n1]+x[n2]}/2 ;find median
29 c=DOT(one,x,n)        ;find mean
30 xmn[ns]=c/on
31 LIST(ns,xmn[ns],xmed[ns]) ;list sample number and mean and median
32 IF(ns.eq.nsamps,density) ;finished all samples?
33 ns=ns+1               ;no
34 GOTO(startloop)
35 ;
36 ;density              ;yes
37 ;
38 DENSITY(xmed,nsamps,rbins,npts,kernel,ymed,fymed)
39 DENSITY(xmn,nsamps,rbins,npts,kernel,ymn,fymn)
40 y=<ymed,ymn>           ;prepare to use PLOTK
41 fy=<fymed,fymn>
42 LABEL(y)= ' '
43 LABEL(fy)='Density for Mean and Median (n=#n#)'
44 type=<2,33>
45 PLOTK(y,fy,npts,2,type)
46 PROMPTON              ;turn prompt back on

```

Example B.3 ARRANGING CALCULATIONS TO SAVE TIME

Every time a command is received by TIMESLAB, it must be interpreted, a process that takes a small amount of time. In fact, for a command that does not perform a large number of calculations, more time is spent interpreting the command than doing the calculations. This is particularly important in macros that have looping constructs, since an IF command whose condition is true causes TIMESLAB to rewind the file and then start reading through it looking for its target line.

To illustrate this, consider a macro that will generate 50 samples of size 10 from a uniform population and find the sample mean for each. The following macro does this using both a loop and a much faster way. The slow way takes 15 seconds on a standard PC/AT, while the fast way takes 0.5 second. This is an extreme example, but it is important to try to organize calculations as much as possible to reduce the amount of time required to perform a given task.


```

1 ;;
2 ;;   SLOW.MAC: This macro illustrates how some sequences of
3 ;;           commands can be incorporated into a single command.
4 ;;           We'll generate 50 samples of 10 uniforms each
5 ;;           in two ways: (1) in a loop (very slow), and
6 ;;           (2) in a single command (very fast).
7 ;;
8 ;;   Input : none
9 ;;
10 PAUSE
11 ;start
12 CLS      ;clear screen
13 ;;
14 ;;   Slow Way (this takes a while, I'll keep you informed
15 ;;           as to which sample we're doing)
16 ;;
17 means=LINE(50,0,0)
18 ns=1
19 seed=12345.
20 x=wn(seed,10)
21 TIME(t1)
22 PROMPTOFF
23 ;
24 ;startloop
25 ;
26 x=WN(0,10,2)
27 x=SUBMNS(x,10,1,means[ns],0)
28 LIST(ns)
29 IF(ns.eq.50,endloop)
30 ns=ns+1
31 GOTO(startloop)
32 ;
33 ;endloop
34 ;
35 TIME(t2)
36 time=t2-t1
37 LIST(time)
38 PAUSE
39 HIST(means,50,5)
40 CLS
41 PROMPTON
42 ;;
43 ;;   FAST WAY:
44 ;;
45 ;fast
46 seed=12345.
47 x=wn(seed,10)
48 TIME(t1)
49 x=WN(0,500,2)
50 x=TRANS(x,10,50)
51 x=SUBMNS(x,500,50,means,0)
52 TIME(t2)
53 time=t2-t1
54 LIST(time)

```

```

55 FAUSE
56 HIST(means,50,5)

```

Example B.4 USING THE SPEAKER

The **SPEAKER** command can be useful in a macro to alert the user that a macro has ended or else that a certain part of the macro has finished. Just for fun, we include the following macro that will produce the theme from "Ode to Joy" on the speaker when invoked.

```

1 ;;
2 ;;   ODE.MAC: macro to play 'Ode to Joy' on the speaker.
3 ;;           To speed it up or slow it down, just
4 ;;           decrease or increase the value of nh.
5 ;;
6 ;;
7 FAUSE
8 ;start
9 nh=30
10 nh2=2*nh
11 f=<659,659,698,784,784,698,659,587,523,523>
12 f=<f,587,659,659,587,587,659,659,698,784,784>
13 f=<f,698,659,587,523,523,587,659,587,523,523>
14 f=<f,587,587,659,523,587,698,659,523,587,698>
15 f=<f,659,587,523,587,784,659,659,698,784,784>
16 f=<f,698,659,587,523,523,587,659,587,523,523>
17 t=<nh,nh,nh,nh,nh,nh,nh,nh,nh,nh>
18 t=<t,nh,nh,nh,nh,nh2,nh,nh,nh,nh,nh>
19 t=<t,nh,nh,nh,nh,nh,nh,nh,nh,nh,nh2>
20 t=<t,nh,nh,nh,nh,nh,nh,nh,nh,nh,nh>
21 t=<t,nh,nh,nh,nh,nh2,nh,nh,nh,nh,nh>
22 t=<t,nh,nh,nh,nh,nh,nh,nh,nh,nh,nh2>
23 SPEAKER(f,t,60)

```

Another interesting use of the speaker is to "play an array," that is, to take the successive elements of an array and have the speaker sound in a frequency proportional to the magnitudes of the elements. In the **PLAY** macro, we form the plot of an array and then while the plot is on the screen, we play it on the speaker. The user must specify the smallest and largest frequencies wanted.

```

1 ;;
2 ;;   PLAY.MAC: macro to 'play an array,' i.e. to take the first
3 ;;           n elements of the array x and make the speaker
4 ;;           sound at a frequency proportional to each element.
5 ;;
6 ;;   INPUT: x, n (array to be played and how many elements to play)
7 ;;           fmin, fmax (minimum and maximum frequencies to use;
8 ;;           these should be between 25 and 20,000)

```

```

9 ;;          time (number of hundredths of seconds each tone is
10 ;;          to play; this should be between 1 and 100)
11 ;;
12 PAUSE
13 ;start
14 PLOTOM      ;go to graphics mode
15 BATCHOM     ;no pause after PLOT
16 PLOT(x,n)   ;plot data
17 MAXMIN(x,n,xmax,im,xmin,im) ;convert range of data to [fmin,fmax]
18 f=fmin+{(x-xmin)/(xmax-xmin)}*(fmax-fmin)
19 t=LINE(n,time,0) ;form array of times
20 SPEAKER(f,t,n) ;play array
21 BATCHOFF    ;turn batch mode off
22 PLOTOFF     ;go back to text mode

```

Example B.5 DRAWING A CIRCLE

Interesting effects can be created on a graphics screen by using the **PLOT2** command with the option to connect the two arrays being plotted. For example, the following macro will draw a solid circle on the screen. One interesting aspect of this is the fact that in order to get the circle to appear actually round, we had to experiment with the scaling factors in the **PLOT2** command as the pixels on the screen are not square. In fact, on most monitors, the ratio of the height to width of a pixel is approximately 12/5.

```

1 ;;
2 ;;  CIRCLE.MAC: macro to draw a circle on a graphics screen.
3 ;;
4 ;;  INPUT: none
5 ;;
6 ;;  OUTPUT: none
7 ;;
8 PAUSE
9 ;start
10 x=LINE(480,-1,1,1) ;x is -1,...,1 (use 480 so every column of
11 ;                  ;pixels gets turned on)
12 y1=(1-x*x)^.5      ;y1 is top half of circle
13 y2=-y1             ;y2 is reflection of y1
14 LABEL(y1)= ' '
15 LABEL(y2)= ' '
16 LABEL(x)= ' '
17 ;
18 ;  Plot on a scale so that the circle looks round.
19 ;
20 PLOT2(y1,y2,480,480,3,3,x,-1.75,1.75,-1,1)

```

Example B.6	SUPERIMPOSING PLOTS
--------------------	----------------------------

An important use of graphics in TIMESLAB is to superimpose many plots on the same screen. In this example we describe three macros that use this ability.

The HAT2 Macro

The HAT2 macro was used to form the top graph in Figure B.3. The macro calculates the $(n \times n)$ hat matrix

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$$

for the $(n \times 3)$, equally spaced quadratic regression matrix

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ 1 & n & n^2 \end{bmatrix}.$$

Then a graph of each of the first $[n/2]+1$ rows (versus their column numbers) is produced on the screen. The only input required from the user is the value of n . In Figure B.3 we used $n = 40$.

The macro also illustrates how labels can be added to a screen using the LABEL command.

```

1 ;;
2 ;;   HAT2.MAC: macro to plot the first [n/2]+1 rows of the
3 ;;           (n x 3) quadratic regression matrix
4 ;;
5 ;;           X = <1,x,x^2>, where x=<1,2,...,n>
6 ;;
7 ;;   INPUT:   n   (less than or equal to 80)
8 ;;
9 PAUSE
10 ;start
11 PLOTOM      ;switch to graphics mode
12 BATCHOM    ;don't want to pause between graphs
13 one=LINE(n,1,0) ;first column
14 X=<one>
15 x=LINE(n,0,1) ;second column

```

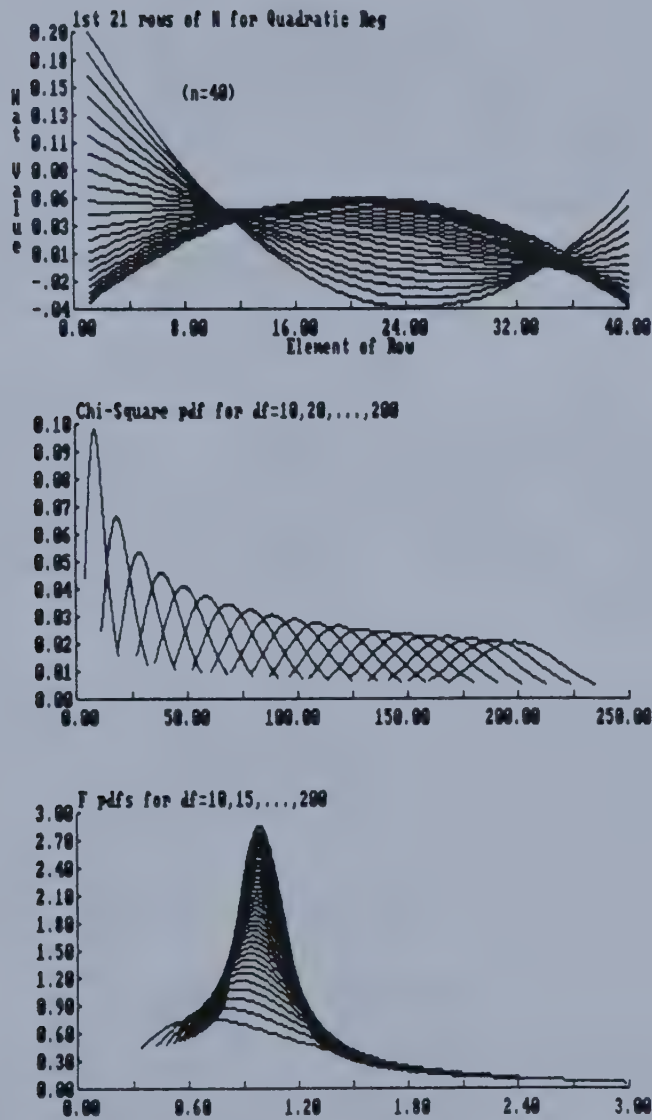


Figure B.3. Three Examples of Superimposing Plots on a Graphics Screen. The Top Graph Is the Output of the HAT2 Macro; the Middle and Bottom Graphs Are from the CHIPLLOT and FPLLOT Macros Respectively.

```

16 X=<X,x>
17 x=x^2                      ;third column
18 X=<X,x>
19 m=3
20 XTX=MMULT(X,X,n,m)         ;X transpose X
21 XTX=MINV(XTX,m,ier)         ;inverse of X transpose X
22 h1=MMULT(X,XTX,n,m,m)
23 XTX=TRANS(X,n,m)
24 HAT=MMULT(h1,XTX,n,m,n)     ;HAT is the hat matrix
25 n2=n^2
26 MAXMIN(HAT,n2,hmax,im,hmin,im) ;need max and min to scale plots
27 i=1                          ;now we'll loop over rows of HAT
28 nb2={n/2}+1                  ;only need to do first 'half' since symmetric
29 ;
30 ;startloop                   ;this is target of implied loop
31 ;
32 i1={i-1}*n+1                 ;i1 and i2 are begin and end of ith row
33 i2=i*n
34 y=EXTRACT(HAT,i1,i2) ;pull off ith row
35 LABEL(y)='Row #i# of H for Quadratic Reg'
36 PLOT(y,n,0,n,hmin,hmax)      ;plot ith row
37 LABEL(150,140,1)='(n=#n#)'   ;put value of n on plot
38 if(i.eq.nb2,endlloop)        ;check if we've done all rows
39 i=i+1                         ;no
40 GOTO(startloop)
41 ;
42 ;endlloop                    ;yes
43 ;
44 LABEL(y,0)='Hat Value'        ;put label on vertical axis
45 LABEL(x,0)='Element of Row'   ;put label on horizontal axis
46 LABEL(cap,0)='1st #nb2# rows of H for Quadratic Reg'
47 GRMENU(0,n,hmin,hmax)        ;put graphics menu on the screen

```

The CHILOT Macro

In this macro, we superimpose the graphs of the pdf of the χ_k^2 distribution:

$$f(x) = \frac{1}{2^{k/2}\Gamma(k/2)} e^{-x/2} x^{(k/2)-1}, \quad x > 0,$$

where Γ is the gamma function

$$\Gamma(\alpha) = \int_0^\infty z^{\alpha-1} e^{-z} dz, \quad \alpha > 0,$$

for degrees of freedom k given by $k = 10, 15, 20, \dots, 200$. Each graph is plotted only from the fifth to the ninety-fifth percentile of the distribution.


```

1 ;;
2 ;;  CHIPLLOT.MAC: macro to superimpose the graphs of the
3 ;;      chi square pdf for degrees of freedom
4 ;;      10 through 200 in steps of 10. Each
5 ;;      plot is from the 5th to the 95th percentile.
6 ;;
7 ;;  INPUT: none
8 ;;
9 PAUSE
10 ;start
11 PLOT0FF          ;switch to graphics mode
12 BATCHON         ;no pause between plots
13 n=10
14 ;
15 ;startloop
16 ;
17 c1=DIST(c,3,1,.05,n)      ;find 5th and 95th percentile
18 c2=DIST(c,3,1,.95,n)
19 x=LINE(100,c1,c2,1)      ;generate values at which to find pdf
20 c=DIST(c,1,100,x,n)      ;find pdf
21 LABEL(x)= ' '
22 LABEL(c)='Chi Square pdf, df=#n#, #n#'
23 PLOT(x,c,100,0,250,0,.10) ;plot it
24 IF(n.eq.200,end)         ;all done?
25 n=n+10                   ;no
26 GOTO(startloop)
27 ;
28 ;end                     ;yes
29 ;
30 LABEL(cap,0)='Chi-Square pdf for df=10,20,...,200'
31 GRMENU(0,250,0,.10)      ;let user FIND or PRINT, etc
32 BATCHOFF                ;return to normal
33 PLOT0FF

```

The FPLLOT Macro

In this macro, we superimpose the graphs of the pdf of the F_{k_1, k_2} distribution:

$$f(x) = \frac{\left(\frac{k_1}{k_2}\right)^{k_1/2} x^{(k_1/2)-1}}{\beta\left(\frac{k_1}{2}, \frac{k_2}{2}\right) \left(1 + \frac{k_1 x}{k_2}\right)^{(k_1+k_2)/2}}, \quad x > 0,$$

where β is the beta function

$$\beta(a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$$

for degrees of freedom k_1 and k_2 given by $n = k_1 = k_2 = 10, 15, 20, \dots, 200$. Each graph is plotted only from the fifth to the ninety-fifth percentile of the distribution.

APPENDIX C

The TIMESLAB Commands

This appendix has two sections. In the first we give an outline of the TIMESLAB commands according to their function. In Section C.2 we describe each command in detail.

C.1. Outline of Commands

The TIMESLAB commands can be loosely grouped into the following categories by their purpose:

COLORS	ESTIMATION	MATRICES	SIMULATION
CORRELATIONS	GRAPHICS	MODELS	SPEAKER
DISTRIBUTION	IO	POLYNOMIALS	SPECTRA
DOS	LISTING	PREDICTION	TRANSFORMS
EDITOR	MACROS	PRINTING	WNTTESTS
ENVIRONMENT	MATH		

COLORS

The colors in the text and graphics modes can be modified via the `TEXT-COLOR` and `COLOR` commands respectively. The colors that are available are listed under these commands in Section C.2.

CORRELATIONS

TIMESLAB has the following commands for manipulating auto- and cross-correlations:

- | | |
|----------|--|
| ARCORR | - Find correlations corresponding to AR parameters. |
| ARCORR2 | - Find correlations corresponding to bivariate AR parameters. |
| ARMACORR | - Find correlations corresponding to ARMA parameters. |
| ARPART | - Find partial correlations corresponding to AR parameters. |
| CORR | - Find sample correlations and/or spectral density for a data set. |
| CORR2 | - Find sample auto- and cross-correlations for bivariate data. |
| CORRAR | - Find AR parameters from correlations. |
| CORRAR2 | - Find AR parameters from bivariate correlations. |
| CORRARMA | - Find ARMA parameters from correlations. |
| CORRMA | - Find MA parameters from correlations. |
| MACORR | - Find correlations corresponding to MA parameters. |
| PARCORR | - Find partial correlations and standardized residual variances from data. |
| PARTAR | - Find partial correlations from AR parameters. |

DISTRIBUTION

TIMESLAB has the following commands for studying the distribution of data:

- | | |
|---------|---|
| DENSITY | - Nonparametric probability density estimation. |
| DIST | - Pdf, cdf, or quantile function of Z , t , χ^2 , or F distribution. |
| HIST | - Calculate and plot histogram. |
| INFQNT | - Calculate and plot informative quantile function. |

DOS

It is possible to issue DOS commands from within TIMESLAB by using the command called DOS.

EDITOR

TIMESLAB has its own built-in text editor (called EDIT) for creating and modifying data or MACRO files.

ENVIRONMENT

The environment under which TIMESLAB operates can be varied by using the following commands:

- BATCHOFF - Cancel BATCHON.
- BATCHON - Henceforth there will be no pause after a plot is produced (this can be used in conjunction with PSON to produce a printed copy of an unattended session).
- CLEAN - Delete a variable or variables.
- CLS - Clear screen.
- COLOR - Set colors in the graphics mode.
- ECHO - Same as Ctrl-Prtsc, that is, toggles printer echo.
- OVEROFF - Henceforth any command that would cause the replacement of an existing variable will cause TIMESLAB to ask permission before doing the replacement.
- OVERON - Any command that would replace an existing variable will do so.
- PLOTOFF - Switch to the text mode.
- PLOTON - Switch to the graphics mode.
- PLOTSIZE - Set graphics parameters (size and location of plots, number of tic marks, and number of characters in tic mark numerical labels).
- PRINTER - Issue commands to the printer.
- PRINTSEL - Tell TIMESLAB what kind of printer is being used.
- PSOFF - Cancel PSON.
- PSON - Henceforth any plot on screen will also go to the printer.
- RECORD - Keep a record of the session on a file.
- RESTART - Equivalent to QUIT and then restarting TIMESLAB.
- TEXTCOLOR - Set colors in the text mode.

ESTIMATION

TIMESLAB has the following commands for estimation:

- | | |
|----------|--|
| ARMASEL | - Subset univariate ARMA fitting and estimation. |
| ARSP | - Used with DTAR for AR spectral estimation. |
| ARSP2 | - Use with CORR2 and CORRAR2 to do bivariate AR spectral estimation. |
| ARSPCB | - Used to find simultaneous confidence bands for AR spectra. |
| ARSPPEAK | - Used to find estimates of peak frequencies for AR spectra. |
| COEFFCSD | - Find asymptotic standard errors for ARMA parameter estimates. |
| CORR | - Univariate sample correlations and spectral density. |
| CORR2 | - Auto- and cross-correlations for bivariate data. |
| CORRAR | - Use with CORR to do univariate AR modeling. |
| CORRAR2 | - Use with CORR2 to do bivariate AR modeling. |
| CROSSP | - Bivariate nonparametric spectral estimation. |
| DTAR | - Univariate AR parameter estimation. |
| DTARMA | - Exact MLE's for ARMA parameters. |
| SEASEST | - Approximate MLE's for Box-Jenkins type models. |
| WINDOW | - Univariate nonparametric spectral estimation. |

GRAPHICS

TIMESLAB has the following high resolution graphics commands:

- | | |
|--------|--|
| CLS | - Erase a graphics screen. |
| ERASE | - Erase part of a graphics screen. |
| FIND | - TIMESLAB has an interactive locator/labeler utility for graphics screens called FIND. For information about FIND, see Section B.5.5. |
| GRMENU | - Place the graphics menu on a graphics screen. |
| HIST | - Calculate and plot histogram for an array. |
| INFQNT | - Calculate and plot informative quantile plot for an array. |

- LABEL - Define a new plotting label for an array to be plotted or place a character string onto an existing screen. For information about labeling plots, see Section B.5.7.
- PLOT - General plotting of Y versus X .
- PLOT2 - Superimpose 2 Y versus X plots.
- PLOTCSF - Plot a spectral array with vertical axis not on log scale, horizontal axis from 0 to .5 cycle per unit time.
- PLOTK - Superimpose K Y versus X plots.
- PLOTOFF - Switch to the text mode.
- PLOTON - Switch to the graphics mode.
- PLOTSP - Plot a spectral array with vertical axis on log scale, horizontal axis from 0 to .5 cycle per unit time.
- RESCREEN - Retrieve previously saved graphics screen or screens and put back on screen.
- SAVESC - Save a compressed image of a graphics screen to a file.

IO

The TIMESLAB commands for reading data from a disk file and writing data to a disk file from within TIMESLAB are called READ and SAVE respectively.

LISTING

TIMESLAB has the following commands for producing textual displays on the screen:

- INFO - Display information about all quantities that have been defined.
- LIST - Display a set of scalar variables or the elements of an array.
- LISTM - Display the elements of a two- or three-dimensional array.
- LISTSP - Display the elements of a spectral array and the corresponding frequencies.

MACROS

TIMESLAB can be used as an interpreted programming language by forming a disk file (via the EDIT command) containing a series of TIMESLAB commands, and then telling TIMESLAB to execute the commands in the file by using the MACRO command. Such a file is referred to as a macro.

The execution of a macro can be interrupted by striking the F9 function key. It can be restarted by issuing the MACRO command with no argument. While the macro has been interrupted, other commands can be issued from the keyboard. For more information and examples, see Section B.8.

The following commands are used with macros:

- ABORTOFF - Henceforth anytime a macro encounters an error, it will continue executing the commands in the macro.
- ABORTON - Henceforth anytime a macro encounters an error it will stop.
- DELAY - The macro pauses for a specified amount of time before resuming execution.
- PAUSE - When this command is encountered, TIMESLAB issues a message to the screen and awaits the user's response. If the user precedes this command with comments that start with two semicolons (which causes them to be displayed on the screen), the user can break the macro, provide information that the macro needs, and then restart the macro.
- PROMPTOFF - Henceforth the commands being executed will not be displayed on the screen.
- PROMPTON - Henceforth each command being executed will be displayed on the screen.
- SINGLEOFF - Cancel SINGLEON.
- SINGLEON - Henceforth the macro waits for the user to strike a key before executing each command.
- SPEAKEROFF - Henceforth the speaker will not sound as each command is executed.
- SPEAKERON - Henceforth the speaker will sound as each command is executed.
- TIME - Determines elapsed time since midnight. TIME can be used to time how long a series of commands takes to execute.

MATH

TIMESLAB has the following commands for doing mathematical operations:

- | | |
|-------|--|
| TYPE2 | - This is actually a set of commands for doing arithmetic operations such as adding, subtracting, multiplying, dividing, and exponentiating reals, integers, and arrays as well as simple assignment statements. |
| ABS | - Absolute value function. |
| ARDT | - Generate values of a difference equation. |
| ARDT2 | - Generate values of a bivariate difference equation. |
| BINOM | - Find binomial coefficients or probabilities. |
| COS | - Cosine function. |
| CUM | - Cumulative sums or averages of an array. |
| DIST | - Pdf, cdf, quantile of Z , t , χ^2 , F distributions. |
| EXP | - Exponential function. |
| LINE | - Form a line. |
| LOGE | - Natural log function. |
| POLY | - Evaluate a polynomial. |
| SIN | - Sine function. |
| SORT | - Sort an array. |

For information on what vector and matrix commands are available, see the MATRICES section that follows.

MATRICES

TIMESLAB has the following commands for doing vector and matrix operations.

- | | |
|---------|---|
| DOUBLE | - Extend an array by symmetry. |
| EIG | - Find eigenvalues and (optionally) eigenvectors. |
| EXTRACT | - Use to extract parts of arrays. |
| GS | - Gram-Schmidt decomposition. |
| LISTM | - List the elements of a matrix or three-dimensional array. |
| MCHOL | - Modified Cholesky decomposition. |

MDEL	- Delete some rows of a matrix.
MINV	- Invert a matrix.
MMULT	- Multiply matrices.
REVERSE	- Form an array by reversing the order of the elements of another array.
SWEEP	- Sweep a matrix.
TOEPL	- Form a Toeplitz matrix.
TRANS	- Transpose a matrix.
TYPE4	- Use to concatenate matrices.

MODELS

The TIMESLAB commands for studying models fall into the following areas:

CORRELATIONS ESTIMATION PREDICTION SIMULATION
SPECTRA

For the commands in each of these areas, see the individual area descriptions in this section.

POLYNOMIALS

TIMESLAB has the following commands for studying polynomials:

INVPOLY	- Find the coefficients of the reciprocal of a polynomial.
MULTPOLY	- Find the coefficients of the product of two polynomials.
POLY	- Evaluate a polynomial.
POLYROOTS	- Find the roots of a polynomial given its coefficients.
ROOTSPOLY	- Find the coefficients of polynomial given its roots.

PREDICTION

TIMESLAB has the following commands for doing prediction:

ARMAPRED	- Prediction for ARMA models.
DTFORE	- Iterated AR forecasting.
EXTEND	- Inverse operation to differencing.
SEASPREDE	- Box-Jenkins forecasting.

PRINTING

TIMESLAB has the following commands for getting printed output:

- ECHO - Same as Ctrl-Prts, that is, toggles printer echo.
- LIST - The display of variables can be directed to the printer via the LIST command.
- PAGE - Position printer to top of the next page.
- PRINT - Print a set of scalar variables or the values of an array.
- PRINTER - Send control sequences to printer.
- PRINTSEL - Tell TIMESLAB what type of printer is being used.
- PSON - Henceforth any plot on screen will go to the printer (cancelled by PSOFF).
- RECORD - Keep a record of a session on the printer.

SIMULATION

TIMESLAB has the following commands for doing simulation:

- ARDT - Simulate data from an AR process.
- ARDT2 - Simulate data from a bivariate AR process.
- ARMA - Simulate data from an ARMA process.
- MADT - Simulate data from an MA process.
- WN - Simulate data from a white noise process.

SPEAKER

TIMESLAB has a command called SPEAKER which allows the user to control the PC's speaker. This can be useful in macros to alert the user that certain parts of the macro have been reached.

SPECTRA

TIMESLAB has the following commands for spectral analysis:

- ARMASP - Spectra for ARMA process.
- ARSP - Spectra for AR process.
- ARSP2 - Spectra for bivariate AR process.
- ARSPCB - Simultaneous confidence bands for AR spectra.

ARSPPEAK	- Estimate frequency of peak in AR spectra.
CORR	- Periodogram and/or correlations of univariate data.
CROSSP	- Nonparametric cross-spectral analysis for bivariate data.
CUMSP	- Cumulative spectra.
FFT	- Fast Fourier transform.
MASP	- Spectra for MA process.
POLAR	- Find amplitude and phase of complex array.
WINDOW	- Nonparametric spectral estimation for univariate data.

TRANSFORMS

TIMESLAB has the following commands for transforming data:

ARDT	- Univariate difference equation.
ARDT2	- Bivariate difference equation.
ARFILT	- Apply AR filter.
DIFF	- Find difference of data.
DIVSDS	- Calculate and divide seasonal standard deviations.
EXP	- Exponential function.
FFT	- Fast Fourier transform.
FILT	- General linear filtering operation.
LOGE	- Natural log.
REG	- Regression analysis.
SORT	- Sort an array.
SUBMNS	- Calculate and subtract means.

WNTTESTS

TIMESLAB has two commands for testing for white noise:

BARTTEST	- Bartlett's test.
QTEST	- Portmanteau test.

C.2. Detailed Description of Commands

In this section we describe the purpose, format(s), input and output arguments (if any) for each of the commands as well as any remarks about special features of the command. In Appendix E of this book is a short list of all of the commands in TIMESLAB together with where in the book the command is introduced. Note that in the description of the format of a command, any argument that is placed in square brackets, such as in

```
y=DIST(name,iopt,n,x[,df1,df2])
```

is an optional argument, that is, sometimes it is needed and sometimes it is not.

ABORTOFF Command

Purpose:

Cancel effect of ABORTON; that is, henceforth macros that encounter an error during their execution will continue running.

Format:

```
ABORTOFF
```

ABORTON Command

Purpose:

Henceforth any error encountered during execution of a macro will cause the macro to stop at that line.

Format:

```
ABORTON
```

Remarks:

1. ABORTON is cancelled by ABORTOFF.
2. ABORTON is in effect when TIMESLAB is started.
3. If a macro is aborted, one can usually fix the error and then restart the macro at the line causing the error by issuing the command MACRO.

ABS Command

Purpose:

To find the absolute value of a scalar or of the first n elements of an array.

Format:

1. $Y=ABS(X)$
2. $Y=ABS(X,n)$

Input:

- | | |
|-----|---|
| X | - In the first form a real scalar and in the second form an array of at least n elements. |
| n | - An integer containing the number of elements of X for which to find the absolute value. |

Output:

- | | |
|-----|---|
| Y | - In the first form a real scalar, and in the second form an array of n elements. |
|-----|---|

ARCORR Command

Purpose:

Calculate the variance $R(0)$ and autocorrelations $\rho(1), \dots, \rho(M)$ of an $AR(p, \alpha, \sigma^2)$ process.

Format:

$\text{rho}=\text{ARCORR}(\text{alpha},p,\text{rvar},M,R0,\text{ier})$

Input:

- | | |
|----------------|--|
| alpha | - Array of length p containing coefficients α . |
| p | - Integer containing order p (>0). |
| rvar | - Real scalar containing error variance σ^2 (>0). |
| M | - Integer containing the number of autocorrelations to calculate (≥ 0). |

Output:

- R0 - Real scalar containing variance $R(0)$ of process.
- ier - Integer variable indicating whether or not the AR process is stationary (0 means yes, anything else means no).
- rho - Array of length M containing autocorrelations $\rho(1), \dots, \rho(M)$.

Remarks:

1. If $ier \neq 1$, then R0 and rho are not calculated.
2. If $M=0$, then R0 is calculated but rho is not.
3. $Workspace=M+p$.

ARCORR2 Command

Purpose:

To find the auto- and cross-correlations for a bivariate AR process.

Format:

ARCORR2(A,p,sigma,M,R10,R20,rho120,rho1,rho2,rho12,rho21,ier)

Input:

- A - Array of length $4 \times p$ containing AR coefficient matrices (the first two elements are the first column of first matrix, next two are second column, etc.).
- p - Integer (> 0) containing AR order.
- sigma - Array of length 4 containing error covariance matrix.
- M - Integer containing the number of correlations to calculate.

Output:

- R10 - Real scalar variable containing the variance of the first univariate process.
- R20 - Real scalar variable containing the variance of the second univariate process.
- rho120 - Real scalar variable containing the cross-correlation of lag 0.
- rho1 - Array of length M containing the autocorrelations of lags 1 through M for the first univariate series.

- rho2 - Array of length M containing the autocorrelations of lags 1 through M for the second univariate series.
- rho12 - Array of length M containing the cross-correlations of lags 1 through M .
- rho21 - Array of length M containing the cross-correlations of lags -1 through $-M$.
- ier - Integer variable which is 0 if the process is judged to be stable (all zeros of determinantal polynomial outside the unit circle) and 1 if not.

Remarks:

1. If $ier=1$, then the other output variables are not returned.
2. $Workspace=5(M+1)+9p+2$.

ARDT Command

Purpose:

1. Generate a realization of length n from a Gaussian $AR(p, \alpha, \sigma^2)$ process X or
2. Generate successive values of a general difference equation:

$$X(t) = \begin{cases} e(i), & i = 1, \dots, p \\ e(i) - \sum_{j=1}^p \alpha(j)X(i-j), & i = p+1, \dots, n. \end{cases}$$

Format:

1. $X=ARDT(alpha, p, rvar, seed, n, ier, R0)$
2. $X=ARDT(alpha, p, n, e)$

Input:

- alpha - Array of length p containing coefficients α .
- p - Integer containing order p (>0).
- rvar - Real scalar containing error variance σ^2 (>0).
- seed - Real scalar containing the seed for the random number generator.
- n - Integer ($>p$) containing the length of the realization.

- e** - Array of length n containing the values of e for the general difference equation.

Output:

- ier** - Integer variable indicating whether or not the AR process is stationary (0 means yes, anything else means no).
- R0** - Real variable containing the variance $R(0)$ of the process.
- X** - Array of length n containing realization.

Remarks:

1. If $ier=1$, then $R0$ and X are not calculated.
2. $Workspace=n+p+p^2$ for form 1, and $n+p$ for form 2.

ARDT2 Command

Purpose:

1. Generate a realization of length n from a bivariate Gaussian $AR(p, \mathbf{A}, \mathbf{\Sigma})$ process \mathbf{X} or
2. Generate successive values of a general difference equation:

$$\mathbf{X}(t) = \begin{cases} \mathbf{e}(i), & i = 1, \dots, p \\ \mathbf{e}(i) - \sum_{j=1}^p \mathbf{A}(j)\mathbf{X}(i-j), & i = p+1, \dots, n. \end{cases}$$

Format:

1. $X=ARDT2(A,p,sigma,seed,n,ier)$
2. $X=ARDT2(A,p,n,e,ier)$

Input:

- A** - Array of length $4*p$ containing AR coefficient matrices (the first two elements are the first column of first matrix, next two are second column, etc.).
- p** - Integer containing AR order.
- sigma** - Array of length 4 containing error covariance matrix.
- seed** - Real scalar containing the seed for the random number generator.

- n** - Integer ($>p$) containing the length of the realization.
- e** - Array of length $2*n$ containing the values of **e** for the general difference equation.

Output:

- ier** - Integer variable indicating whether or not the difference equation is stable (0 means yes, anything else means no).
- X** - Array of length $2*n$ containing realization.

Remarks:

1. If **ier**=1, then **X** is not calculated.
2. The first two elements of **X** are **X**(1), the next two are **X**(2), and so on.
3. **Workspace**= $4(n+p+1)$ for form 1, and $4(n+p)$ for form 2.

ARFILT Command

Purpose:

To apply an $AR(p, \alpha, \sigma^2)$ filter to an $(n \times m)$ matrix **X**, that is, to multiply **X** by $\mathbf{D}^{-1/2}\mathbf{L}^{-1}$ where $\mathbf{V} = \mathbf{LDL}^T$ is the Cholesky decomposition of the $(n \times n)$ covariance matrix of the AR process.

Format:

Y=ARFILT(alpha,p,rvar,X,n,m,ier)

Input:

- alpha** - Array of length **p** containing coefficients α .
- p** - Integer containing order p (>0).
- rvar** - Real scalar containing error variance σ^2 (>0).
- X** - Array containing the $(n \times m)$ matrix to be filtered.
- n,m** - Integers containing the size of **X**.

Output:

- Y** - Array of length **nm** containing the filtered version of **X**.
- ier** - Integer variable indicating whether or not the AR process is stationary (0 means yes, anything else means no).

Remarks:

1. If $\text{ier} \neq 0$, then Y is not formed.
2. $\text{Workspace} = 2nm + p^2 + 2p$.

ARMACORR Command

Purpose:

Calculate the variance $R(0)$ and autocorrelations $\rho(1), \dots, \rho(M)$ of an $\text{ARMA}(p, q, \alpha, \beta, \sigma^2)$ process given its parameters.

Format:

```
rho=ARMACORR(alpha,beta,p,q,rvar,M,R0,ier)
```

Input:

- | | | |
|-------|---|---|
| alpha | - | Array of length p containing coefficients α . |
| beta | - | Array of length q containing coefficients β . |
| p | - | Integer containing order p (>0). |
| q | - | Integer containing order q (>0). |
| rvar | - | Real scalar containing error variance σ^2 (>0). |
| M | - | Integer ($\geq \max(p, q)$) containing the number of autocorrelations to calculate. |

Output:

- | | | |
|-----|---|--|
| R0 | - | Real scalar containing variance $R(0)$ of process. |
| ier | - | Integer variable indicating whether or not the ARMA process is stationary (0 means yes, anything else means no). |
| rho | - | Array of length M containing autocorrelations $\rho(1), \dots, \rho(M)$. |

Remarks:

1. If $\text{ier}=1$, then $R0$ and rho are not calculated.
2. If $M=0$, then $R0$ is calculated but rho is not.
3. $\text{Workspace} = p + q + M + 1 + 2(\max(p, q) + 1)$.

ARMADT Command

Purpose:

Generate a realization of length n from a Gaussian ARMA($p, q, \alpha, \beta, \sigma^2$) process X .

Format:

`X=ARMADT(alpha,beta,p,q,rvar,seed,n,M,rho,r0,ier)`

Input:

- `alpha` - Array of length p containing coefficients α .
- `beta` - Array of length q containing coefficients β .
- `p` - Integer containing order p (>0).
- `q` - Integer containing order q (>0).
- `rvar` - Real scalar containing error variance σ^2 (>0).
- `seed` - Real scalar containing the seed for the random number generator.
- `n` - Integer ($>\max(p, q)$) containing the length of the realization.
- `M` - Integer containing the number of autocorrelations to calculate ($>\max(p, q)$).

Output:

- `rho` - Array of length M containing autocorrelations $\rho(1), \dots, \rho(M)$.
- `R0` - Real scalar containing variance $R(0)$ of process.
- `ier` - Integer variable indicating whether or not the ARMA process is stationary (0 means yes, anything else means no).
- `X` - Array of length n containing realization.

Remarks:

1. If `ier=1`, then `rho`, `R0`, and `X` are not calculated.
2. `Workspace=p+q+r2+5M+2n`, where `r=max(p, q+1)`.

ARMAPRED Command

Purpose:

Find minimum mean square error linear predictors and (optionally) prediction standard errors for an ARMA($p, q, \alpha, \beta, \sigma^2$) process X .

Format:

ARMAPRED($X, n, \alpha, \beta, p, q, rvar, t1, t2, h1, h2, XP, ier [, SE]$)

Input:

- X - Array of length n containing the realization to be used in the prediction.
- n - Integer containing the length of the realization (>0).
- α - Array of length p containing coefficients α .
- β - Array of length q containing coefficients β .
- p - Integer containing order p (>0).
- q - Integer containing order q (>0).
- $rvar$ - Real scalar containing error variance σ^2 (>0).
- $t1, t2$ - Integers ($1 \leq t1 \leq t2 \leq n$) specifying the range of memories to be used (see Remark 1 below).
- $h1, h2$ - Integers ($1 \leq h1 \leq h2$) specifying the horizons to be used (see Remark 1 below).

Output:

- ier - Integer variable indicating whether or not the ARMA process is stationary (0 means yes, anything else means no).
- XP - Array of length $(t2 - t1 + 1)(h2 - h1 + 1)$ containing predictors (see Remark 1 below).
- SE - If this argument is present, it contains the prediction standard errors for the predictors in the array XP .

Remarks:

1. The memory- t , horizon- h predictor and prediction standard error of $X(t+h)$ given $X(1), \dots, X(t)$ are found for t between t_1 and t_2 , for h between h_1 and h_2 , and placed into the arrays XP and SE (SE is only formed if present in the argument list). The values for $t=t_1$ come first, then come the values for $t=t_1+1$, and so on.
2. $Workspace=n+6r^2+6r+j*npreds$, where $r=\max(p,q+1)$ and j is 1 if the SE argument is not present, and j is 2 if it is present.

ARMASEL Command

Purpose:

Do a stepwise ARMA analysis of data $X(1), \dots, X(n)$.

Format:

ARMASEL($X, n, M, s, k_1, k_2, kopt, pval, p, q, alpha, beta, rvar, ier$)

Input:

- | | |
|------------|--|
| X | - Array of length n containing the time series to be analyzed. |
| n | - Integer containing the length of the data to be analyzed. If n is negative, no output is displayed and the sample size is taken as the absolute value of n . |
| M, s | - Integers determining the maximum AR and MA lags that can be included in the model. An AR model of length s , $s \leq M$ is fit to X and then the largest AR and MA lags that can be included are $M-s$. Thus $1 \leq M-s \leq 40$. |
| k_1, k_2 | - Integers giving the number of AR and MA lags that can be included in the model ($0 \leq k_1, k_2 \leq M-s$, see Remark 1 below). |
| $kopt$ | - Integer determining whether ARMASEL should choose the number of lags and what they are ($kopt=0$), whether the user will specify the number of lags and ARMASEL will choose what they are ($kopt>0$), or whether the user will specify both the number of lags and what they are and ARMASEL will only estimate the coefficients ($kopt<0$). See Remark 1 below for details. |
| $pval$ | - Real scalar specifying the p -value for inclusion at each step of the stepwise procedure ($0 < pval < 1$). Note that a small value for $pval$ makes it easier for a lag to be included. |

- alpha** - (If $kopt < 0$ and $k1 > 0$) Array of length $k1$ specifying the AR lags to be forced into the model.
- beta** - (If $kopt < 0$ and $k2 > 0$) Array of length $k2$ specifying the MA lags to be forced into the model.

Output:

- p** - Integer variable containing the order of the AR part of the chosen ARMA model. Thus p is the maximum AR lag chosen.
- q** - Integer variable containing the order of the MA part of the chosen ARMA model. Thus q is the maximum MA lag chosen.
- alpha** - (If $p > 0$) Real array of length p containing the coefficients for the AR lags that were chosen and zeros for those not chosen.
- beta** - (If $q > 0$) Real array of length q containing the coefficients for the MA lags that were chosen and zeros for those not chosen.
- rvar** - Real scalar containing the estimate of the error variance.
- ier** - Integer indicating if any numerical errors were encountered (0 means no, 1 means the matrix being swept becomes singular).

Remarks:

1. If $kopt=0$, then the $k1$ AR lags and $k2$ MA lags out of the M -s possible lags having highest correlation in absolute value with $X(t)$ are possible contenders for inclusion in the model and ARMASEL will continue adding and deleting lags until the stepwise procedure says to stop.
2. If $kopt=j>0$, then the same lags as in Remark 1 are contenders for inclusion, but ARMASEL will force into the model the "best" j lags, that is, the lags chosen by the stepwise procedure.
3. If $kopt=j<0$, then the $-j = k1 + k2$ lags inputted in the arrays **alpha** and/or **beta** are forced into the model.
4. $Workspace=2M+4(M-s)^2+700$.

ARMASP Command

Purpose:

Calculate the spectral density of an $ARMA(p, q, \alpha, \beta, \sigma^2)$ process.

Format:

`f=ARMASP(alpha,beta,p,q,rvar,Q)`

Input:

- `alpha` - Array of length `p` containing coefficients α .
- `beta` - Array of length `q` containing coefficients β .
- `p` - Integer containing order p (>0).
- `Q` - Integer ($>\max(p,q)$) containing the number of frequencies between 0 and 1 at which to calculate the spectral density. The greatest prime factor of `Q` can be at most 23, and the product of its square-free prime factors can be at most 210.
- `rvar` - Real scalar containing error variance σ^2 (>0).
- `Q` - Number of frequencies between 0 and 1 at which to calculate spectra (the greatest prime factor of `Q` must be ≤ 23 , and the product of its square-free prime factors must be ≤ 210).

Output:

- `f` - Array of length $m = [Q/2] + 1$ containing the values of the spectral density at the frequencies $(j-1)/Q$, $j = 1, \dots, m$.

Remark:

1. `Workspace=2Q`.

ARPART Command

Purpose:

Calculate the partial autocorrelations of an $AR(p, \alpha, \sigma^2)$ process given its coefficients. ARPART can also be used to check coefficients for having characteristic polynomial with zeros outside the unit circle.

Format:

`part=ARPART(alpha,p,ier)`

Input:

- `alpha` - Array of length `p` containing coefficients α .
- `p` - Integer containing order p (>0).

Output:

- part** - (If **ier**=0) Array of length **p** containing partial autocorrelations.
- ier** - Integer variable indicating whether the zeros of the characteristic polynomial corresponding to **alpha** are all outside the unit circle (0 means they are, anything else means they are not).

ARSP Command

Purpose:

Calculate the spectral density of an $AR(p, \alpha, \sigma^2)$ process.

Format:

f=ARSP(**alpha**,**p**,**rvar**,**Q**)

Input:

- alpha** - Array of length **p** containing coefficients α .
- p** - Integer containing order p (>0).
- rvar** - Real scalar containing error variance σ^2 (>0).
- Q** - Integer ($>p$) containing the number of frequencies between 0 and 1 at which to calculate the spectral density. The greatest prime factor of **Q** can be at most 23, and the product of its square-free prime factors can be at most 210.

Output:

- f** - Array of length $m = [Q/2] + 1$ containing AR spectra at the frequencies $(j - 1)/Q, j = 1, \dots, m$.

Remark:

1. Workspace=2Q.

ARSP2 Command

Purpose:

Calculate auto- and cross-spectra for a bivariate autoregressive process of order p .

Format:

ARSP2(A,p,sigma,Q,fxx,fyy,fxyr,fxyi)

Input:

- A - Three-dimensional array containing AR coefficient matrices for order p .
- p - Integer containing order p (>0).
- sigma - Array of length 4 containing error covariance matrix.
- Q - Integer containing the number of frequencies between 0 and 1 at which to calculate spectra. ARSP2 does not use the fast Fourier transform so there are no restrictions on the factors of Q .

Output:

- fxx - Array containing autospectra of first series.
- fyy - Array containing autospectra of second series.
- fxyr - Array containing the real part of the cross-spectra.
- fxyi - Array containing the imaginary part of the cross-spectra.

Remarks:

1. fxx, fyy, fxyr, fxyi are each real arrays of length $m = [Q/2] + 1$ containing values for frequencies $(j - 1)/Q$, $j = 1, \dots, m$.
2. p, A, and sigma will be in the correct form if they are obtained from CORRAR2.
3. $\text{Workspace} = 4(p+1) + 4[Q/2] + 1$.

ARSPCB Command

Purpose:

Find confidence bands for the spectral density of an $AR(p, \alpha, \sigma^2)$ process.

Format:

ARSPCB(alpha,p,rvar,n,Q,conf,fl,fu)

Input:

- alpha - Array of length p containing coefficients α .
- p - Integer containing order p (>0).
- rvar - Real scalar containing error variance σ^2 (>0).
- n - Integer containing the length of the realization that was used to estimate the coefficients (>0).
- Q - Number of frequencies between 0 and 1 at which to calculate the bands (the greatest prime factor of Q must be ≤ 23 , and the product of its square-free prime factors must be < 210).
- conf - Real scalar containing the desired confidence level for the bands ($0 < \text{conf} < 1$).

Output:

- fl,fu - Arrays of length $m = [Q/2] + 1$ containing the lower and upper confidence bands each at the frequencies $(j-1)/Q$, $j = 1, \dots, m$.

Remark:

1. Workspace = $5(p+1) + 3(p+1)^2 + 3Q$.

ARSPPEAK Command

Purpose:

To find the frequency corresponding to a peak in the spectral density of an $AR(p, \alpha, \sigma^2)$ process.

Format:

ARSPPEAK(alpha,p,rvar,n,ier,wp,se[,start]).

Input:

- alpha - Array of length p containing coefficients α .
- p - Integer ($1 \leq p \leq 55$) containing the order p .
- rvar - Real scalar containing error variance σ^2 (>0).
- n - Integer containing (if ARSPPEAK is being used for estimation purposes) the length of the realization that was used to estimate the parameters of the process. If the parameters are the true values, let $n=1$.
- start - An optional argument that is a real scalar containing a starting value ($0 < \text{start} < .5$) for the maximum finding procedure. If start is not present, then ARSPPEAK uses the largest relative maximum of the spectral density evaluated at 256 frequencies between 0 and 1 as the starting value.

Output:

- ier - An integer indicating whether or not ARSPPEAK was successful in finding a peak. The possible values of ier are:
 - 0 ARSPPEAK was successful in finding a peak.
 - 1 ARSPPEAK judged that the AR spectral density has no relative maxima.
 - 2 A zero second derivative was encountered.
 - 3 The maximum finder converged to frequency 0 or .5.
 - 4 The maximum finder didn't converge.
- wp - (If ier=0) The peak frequency.
- se - (If ier=0) The standard error of the peak frequency estimator.

Remarks:

1. Using different starting values will allow more than one peak to be found if there are more than one.
2. $\text{Workspace}=3(p+p^2)$.

BARTTEST Command

Purpose:

Perform Bartlett's test for the significance of the maximum deviation of a cumulative spectra from the spectral distribution of white noise.

Format:

1. `pval=BARTTEST(F,Q,n,b)`
2. `pval=BARTTEST(b)`

Input:

- | | |
|----------|--|
| F | - Array of length $[Q/2]+1$ containing the values of a cumulative spectra (usually produced by <code>CUMSP</code>). |
| Q | - Integer containing the number of frequencies at which original spectra was calculated. |
| n | - Integer containing the sample size to be used in the test. |

Output:

- | | |
|-------------|---|
| b | - Real scalar variable containing the value of the test statistic. |
| pval | - Real scalar variable containing the p -value corresponding to the b ; that is, pval is the probability (under the hypothesis of white noise) of observing a deviation of size b or more. |

Remarks:

1. The value of $[Q/2]+1$ must equal the length of **F**.
2. In the second form, **b** is input.

BATCHOFF Command

Purpose:

Cancel the effect of `BATCHON`.

Format:

`BATCHOFF`

Remark:

1. While BATCHON is in effect, all variables are automatically overwritten if required. When BATCHOFF is issued, the OVERON/OVEROFF status takes over again.

BATCHON Command

Purpose:

Put TIMESLAB into "batch mode," that is, TIMESLAB does not pause during lists or plots.

Format:

BATCHON

Remark:

1. BATCHON has the effect of OVERON until BATCHOFF is issued.

BINOM Command

Purpose:

Find binomial coefficients and probabilities.

Format:

1. $x = \text{BINOM}(n, k)$
2. $x = \text{BINOM}(n, p)$
3. $x = \text{BINOM}(n, k, p)$

Remarks:

1. In the first form, the output real scalar x contains the binomial coefficient (n choose k) for the input integers n and k .
2. In the second form, the output array x of length $n+1$ contains the binomial probabilities for $0, 1, \dots, n$ successes for n trials and success probability p where n and p are input and are an integer and a real scalar respectively.

- 3. In the third form, the output real scalar contains the probability that a binomial random variable having *n* trials and success probability *p* is equal to *k*. *n* and *k* are input integers and *p* is an input real scalar.
- 4. `Workspace=n+1` in the second form.

CLEAN Command

Purpose:

Delete variables.

Format:

- `CLEAN` - TIMESLAB asks users about each of the arrays that have been defined, giving them the option of either deleting it, not deleting it, or quitting the prompting.
- `CLEAN(list)` - Each of the variables in the list will be deleted.
- `CLEAN(all)` - All of the variables that have been defined will be deleted.
- `CLEAN(arrays)` - All of the arrays that have been defined will be deleted.
- `CLEAN(reals)` - All of the real scalar variables that have been defined will be deleted.
- `CLEAN(ints)` - All of the integer variables that have been defined will be deleted.
- `CLEAN(chars)` - All of the character variables that have been defined will be deleted.

CLS Command

Purpose:

Clear the screen in either the text or graphics mode.

Format:

`CLS`

COEFFCSD Command

Purpose:

Find the asymptotic standard deviation of estimates of AR, MA, or ARMA processes.

Format:

1. COEFFCSD(coeff,p,n,SD)
2. COEFFCSD(alpha,beta,p,q,n,SDALPH,SDBETA,ier)

Remarks:

1. The first form is used for AR or MA processes. `coeff`, `p`, and `n` are the coefficients, order, and sample size and are input, while `SD` is an output array of length `p` containing the asymptotic standard deviations of the coefficients.
2. The second form is for ARMA processes. `alpha`, `beta`, `p`, `q`, and `n` are the coefficients, orders, and sample size. The output arrays `SDALPH` and `SDBETA` of lengths `p` and `q` are the asymptotic standard errors for the AR and MA coefficients respectively, while the output integer variable `ier` is an error indicator (0 means no error, 1 means a singular matrix was encountered).
3. $\text{Workspace} = p^2 + p$ for the first form, and $2r^2 + r + p + q$ in the second form where $r = 2\max(p, q) + 1$.

COLOR Command

Purpose:

Set the foreground color to be used henceforth in the graphics mode.

Format:

COLOR(kolor)

Input:

- `kolor` - An integer determining the color to be used according to the rules given below.

Remarks:

- 1. On a PC with a color graphics adapter (CGA), the background color is always black, while the foreground color can be any one of the 15 colors in the table given below (plus `kolor=0` which gives a black foreground and thus should not be used). To set the foreground color, enter the command `COLOR(kolor)` where `kolor` is one of the 15 values in the table.
- 2. On a PC with an enhanced graphics adapter (EGA), the foreground color can be either green or blue-green, and the background color can be any one of the 15 colors in the table given below (plus `kolor=0` which gives a black background). To set the foreground color to green or blue-green, enter the command `COLOR(256)` or `COLOR(257)` respectively. Then to set the background color, enter the command `COLOR(kolor)`, where `kolor` is 0 for black or any one of the 15 colors in the table.
- 3. `TIMESLAB` starts with yellow on black for a CGA and green on brown for an EGA.

Foreground Colors in the Graphics Mode

kolor	Color	kolor	Color	kolor	Color
1	Blue	6	Yellow-brown	11	Light blue-green
2	Green	7	White	12	Light red
3	Blue-green	8	Gray	13	Light purple
4	Red	9	Light blue	14	Yellow
5	Purple	10	Light green	15	Bright white

CORR Command

Purpose:

Calculate the variance and/or correlogram and/or sample spectral density of a univariate time series data set.

Format:

`rho=CORR(X,n,M,Q,iopt,R0,f)`

Input:

`X` - Array of length `n` containing data.

- n** - Integer containing the length of the data (>0).
- M** - Integer containing the number of correlations to calculate ($0 \leq M < n$).
- Q** - Integer containing the number of frequencies in $[0,1]$ at which to calculate the sample spectral density. The largest prime factor can be at most 23, while the product of the square-free prime factors can be at most 210.
- iopt** - Integer indicating whether the sample mean should be subtracted before doing the calculations (1 means yes, 2 means no, the array **X** is not affected either way).

Output:

- R0** - Real scalar variable containing the sample variance (always calculated).
- f** - Array of length $m = [Q/2] + 1$ containing the sample spectral density at the frequencies $(j - 1)/Q$, $j = 1, \dots, m$ (only calculated if $Q \geq n + M$).
- rho** - Array of length **M** containing correlations (only calculated if $M > 0$).

Remarks:

1. If $M=0$, then **R0** is calculated but **rho** is not.
2. If $Q=0$, then **f** is not calculated.
3. If both **n** and **M** are positive, then **Q** must be at least $n+M$.
4. $\text{Workspace} = n + 2Q + M$.

CORR2 Command

Purpose:

Calculate sample autocorrelations and cross-correlations for two univariate time series.

Format:

`CORR2(X,Y,n,M,iopt,RX0,RY0,rhoxy0,rhox,rhoy,rhoxy,rhoxyx)`

Input:

- X** - Array of length n containing the first series.
- Y** - Array of length n containing the second series.
- n** - Integer containing the length of the data sets.
- M** - Integer ($<n$) containing the number of lags desired.
- iopt** - Integer indicating whether or not to subtract the sample means before doing the calculations (1 means yes, 2 means no, the arrays **X** and **Y** are not effected in either case).

Output:

- RX0** - Real scalar containing sample variance of the first series (>0).
- RY0** - Real scalar containing sample variance of the second series (>0).
- rhoxy0** - Real scalar containing sample cross-correlation of lag 0.
- rhox** - Array of length M containing autocorrelations of lags $1, \dots, M$ for first series.
- rhoy** - Array of length M containing autocorrelations of lags $1, \dots, M$ for second series.
- rhoxy** - Array of length M containing cross-correlations of lags $1, \dots, M$.
- rhoxyx** - Array of length M containing cross-correlations of lags -1 through $-M$.

Remark:

1. $\text{Workspace}=2n+4M$.

CORRAR Command

Purpose:

Calculate the coefficients and error variance of an $\text{AR}(p, \alpha, \sigma^2)$ process given its variance and correlations. In the first form the order is specified, while in the second form the order is chosen by the CAT criterion.

Format:

1. `alpha=CORRAR(rho,R0,p,rvar)`
2. `alpha=CORRAR(rho,R0,M,n,p,rvar,CAT)`

Input:

- rho** - Array of length p (first form) or M (second form) containing autocorrelations.
- R0** - Real scalar containing sample variance (>0).
- p** - (First form) Integer containing the AR order (>0).
- M** - (Second form) Integer containing the largest order to fit (>0).
- n** - (Second form) Integer containing the sample size.

Output:

- p** - (Second form) Integer variable containing the order chosen.
- rvar** - Real scalar variable containing error variance.
- alpha** - Array of length p containing AR coefficients.
- CAT** - (Second form) Array of length M containing values of CAT for orders $1, \dots, M$.

Remarks:

1. If $p=0$ in the second form, then **rvar** and **alpha** are not formed.
2. $\text{Workspace}=3p$ for the first form, and $3M$ for the second form.

CORRAR2 Command

Purpose:

Calculate bivariate AR parameters given variances and correlations.

Format:

$A = \text{CORRAR2}(RX0, RY0, rhoxy0, rhox, rhoxy, rhoxyx, M, iopt, n, p, SIGMA, CAT, ier)$

$A = \text{CORRAR2}(RX0, RY0, rhoxy0, rhox, rhoxy, rhoxyx, p, sigma, ier)$

Input:

- RX0** - Real scalar containing sample variance of the first series (>0).
- RY0** - Real scalar containing sample variance of the second series (>0).

- `rhoxy0` - Real scalar containing sample cross-correlation of lag 0.
- `rhox` - Array of length M containing autocorrelations of lags $1, \dots, M$ for first series.
- `rhoy` - Array of length M containing autocorrelations of lags $1, \dots, M$ for second series.
- `rhoxy` - Array of length M containing cross-correlations of lags $1, \dots, M$.
- `rhoyx` - Array of length M containing cross-correlations of lags -1 through $-M$.
- `M` - Integer (≤ 200) containing the order to fit (if `iopto` is 1) or the maximum order to fit (if `ioto` is 2).
- `iopto` - Integer indicating what AR order(s) to fit. The following values are available:
 - 1 Fit order M .
 - 2 Fit CAT determined order $\leq M$.
- `n` - Integer containing sample size for use in CAT (>0).

Output:

- `p` - Integer variable containing the AR order that was fit.
- `SIGMA` - Array of length 4 containing error covariance matrix.
- `CAT` - Array of length M containing the CAT criterion for orders $1, \dots, M$.
- `ier` - Integer variable indicating possible errors (0 means no, 1 means yes).
- `A` - Three-dimensional array containing AR coefficient matrices for order p (not formed if $p=0$).

Remarks:

1. The second form can be used when the user knows what order to use.
2. The bivariate Levinson's algorithm is used.
3. `ier=1` indicates either that the data are too ill-conditioned or that the user is trying to fit more parameters than there are observations ($2n$ must be $> 4M+3$).
4. `Workspace=42M+4`.

CORRARMA Command

Purpose:

Calculate the coefficients and error variance of an ARMA($p, q, \alpha, \beta, \sigma^2$) process given its variance and autocorrelations.

Format:

rvar=CORRARMA(rho,R0,p,q,maxit,del,ier,alpha,beta)

Input:

- rho - Array of length $p+q$ containing the autocorrelations of the process.
- R0 - Real scalar containing the variance of the process (>0).
- p - Integer containing order p (>0).
- q - Integer containing order q (>0).
- maxit - Integer containing the maximum number of iterations to allow in Wilson's algorithm (see Remark 1 below).
- del - Real scalar containing convergence criterion (>0).

Output:

- ier - Integer variable containing an error/convergence indicator. The following values are possible:
 - 0 CORRARMA successfully found the ARMA parameters.
 - 1 A singular matrix was encountered trying to find AR parameters.
 - 2 Wilson's algorithm for finding the MA parameters didn't converge.
- alpha - Array of length p containing AR coefficients.
- beta - Array of length q containing MA coefficients.
- rvar - Real scalar variable containing the error variance σ^2 .

Remarks:

1. CORRARMA finds alpha by solving the high-order Yule-Walker equations. This system of equations can be singular in which case ier is set equal to 1 and the output quantities are not formed.

2. If finding **alpha** causes no error, then the correlations of the MA part of the model are calculated and Wilson's algorithm is used to convert them to **beta** and **rvar**. This iterative algorithm is not guaranteed to converge in the **maxit** iterations allowed by the user. If it doesn't converge, **ier** is set equal to 2, and **beta** and **rvar** are not formed. A message is displayed upon nonconvergence.
3. $\text{Workspace} = 14(p+q) + (p+q)^2$.

CORRMA Command

Purpose:

Calculate the coefficients and error variance of an $\text{MA}(q, \boldsymbol{\beta}, \sigma^2)$ process given its autocorrelations.

Format:

`beta=CORRMA(rho,q,R0,maxit,del,ier,rvar)`

Input:

- | | |
|--------------|--|
| rho | - Array of length q containing autocorrelations of lags $1, \dots, q$. |
| q | - Integer containing order q (>0). |
| R0 | - Real scalar containing the variance of the MA process. |
| maxit | - Integer containing the maximum number of iterations to use in Wilson's algorithm (>0). |
| del | - Real scalar containing the convergence criterion to use in Wilson's algorithm (>0). |

Output:

- | | |
|-------------|--|
| ier | - Integer variable indicating whether or not Wilson's algorithm converged (0 means yes, 1 means no). |
| rvar | - Real scalar variable containing the error variance σ^2 of the MA process. |
| beta | - Array of length q containing MA coefficients. |

Remarks:

1. If the algorithm doesn't converge in **maxit** iterations, TIMESLAB displays a message giving the difference in consecutive iterations.

2. If `ier` is 1, then `rvar` and `beta` are not formed.
3. `Workspace=10q`.

COS Command

Purpose:

Calculate cosines of various kinds.

Format:

1. `y=COS(x)`
2. `Y=COS(X,n)`
3. `Y=COS(n,a,p)`

Remarks:

1. In the first form, the output real scalar variable `y` is the cosine of the real scalar input (expressed in radians) `x`.
2. In the second form, the output array `Y` of length `n` contains the cosines of the corresponding values in the input array `X`, which must be expressed in radians.
3. In the third form, the elements of the output array `Y` of length `n` (`n` is an input integer) are given by $Y(i) = a \cos(2\pi(i-1)/p)$, where the real scalars `a` and `p` are input and $p \neq 0$.

CROSSP Command

Purpose:

Calculate windowed cross-spectra for two time series given their cross-correlations.

Format:

`CROSSP(rhox,y,rhoyx,RX0,RY0,rhox0,M,Q,iopt,fxyr,fxyi)`

Input:

`rhox` - Array of length `M` containing cross-correlations of lags $1, \dots, M$.

- rhoyx - Array of length M containing cross-correlations of lags -1 through $-M$.
- RX0 - Real scalar containing sample variance of the first series (>0).
- RY0 - Real scalar containing sample variance of the second series (>0).
- rhoxy0 - Real scalar containing sample cross-correlation of lag 0.
- M - Integer containing the truncation point to be used ($0 < M < Q$, must be even for $iopt=6$).
- Q - Number of frequencies between 0 and 1 at which to calculate spectra (the greatest prime factor of Q must be ≤ 23 , and the product of its square-free prime factors must be ≤ 210).
- iopt - Integer indicating the window to use:
 - 1 Truncated periodogram
 - 2 Bartlett
 - 3 Tukey
 - 4 Parzen
 - 5 Bohman

Output:

- fxyr - Array of length $m = [Q/2] + 1$ containing the real part of the cross-spectra at frequencies $(j - 1)/Q$, $j = 1, \dots, m$.
- fxyi - Array of length $m = [Q/2] + 1$ containing the imaginary part of the cross-spectra at frequencies $(j - 1)/Q$, $j = 1, \dots, m$.

Remark:

1. Workspace= $m+3Q$.

CUM Command

Purpose:

Calculate cumulative sums or averages of the first n elements of an array.

Format:

$Y=CUM(X,n,iopt)$

Input:

- X - Array of length at least n containing the input data.
- n - Integer containing the number of sums or averages to find.
- iopt - Integer indicating whether cumulative sums (iopt=1) or cumulative averages (iopt=2) are desired.

Output:

- Y - Array of length n containing cumulative sums or averages.

Remark:

1. Workspace= n .

CUMSP Command

Purpose:

Calculate cumulative spectra given a spectral array.

Format:

$F = \text{CUMSP}(f, Q)$

Input:

- f - Array of length $m = [Q/2] + 1$ containing the values of a spectral density at the frequencies $(j - 1)/Q$, $j = 1, \dots, m$.
- Q - Integer containing the number of frequencies at which fY was calculated.

Output:

- F - Array of length $m = [Q/2] + 1$ containing values of the cumulative spectra at the frequencies $(j - 1)/Q$, $j = 1, \dots, m$.

Remark:

1. Workspace= $[Q/2] + 1$.

DELAY Command

Purpose:

Cause TIMESLAB to delay execution for a specified amount of time, usually in a macro.

Format:

DELAY(t)

Input:

- t - Real scalar containing the amount of time (in seconds) to delay (>0.).

DENSITY Command

Purpose:

Find a kernel probability density estimator for a data set.

Format:

DENSITY(X,n,rbins,M,kernel,Y,FY)

Input:

- X - Array of length n containing data.
- n - Integer containing the length of X.
- rbins - Real scalar determining the bandwidth to be used (see Remark 1).
- M - Integer containing the number of points at which to estimate the density. If M is negative, then the values at which the density is to be estimated are entered in the array Y, and the number of such values is taken to be the absolute value of M.
- Y - (If M is negative) An array containing the values at which the density is to be estimated. The elements of Y must be in increasing order.
- kernel - Integer containing which kernel to use. The following choices are available:

- 1 Rectangular
- 2 Cosine
- 3 Epanechnikov
- 4 Biweight
- 5 Triangular
- 6 Gaussian
- 7 Parzen

Output:

- Y - Array of length M containing values where density is estimated.
- FY - Array of length M containing values of the density estimator at the points in Y.

Remarks:

1. The bandwidth that is used by the DENSITY command is given by

$$b = \begin{cases} \frac{r_x}{2rbins}, & rbins > 0 \\ 1.75s_x n^{-q}, & rbins = 0, \end{cases}$$

where r_x and s_x are the range and standard deviation of the data respectively, and q is $1/3$ for the rectangular window and $1/5$ otherwise.

2. Workspace= $n+2M$.

DIFF Command

Purpose:

Form an array that contains the differences of another array for a specified lag.

Format:

$Y = \text{DIFF}(n, d, X)$

Input:

- n - Integer containing the length of the data.

- d - Integer containing the lag to use ($0 < d < n$).
- X - Array of length n containing the data to be differenced.

Output:

- Y - Array of length $n-d$ containing the result of the differencing X , that is, the i th element of Y is $Y(i) = X(i+d) - X(i)$, $i = 1, \dots, n-d$.

DIST Command

Purpose:

Calculate gamma function (form 1) or pdf, cdf, or quantiles of Z , t , χ^2 , or F distribution (form 2).

Format:

1. `y=DIST(n,a,b)`
2. `y=DIST(name,iopt,n,x[,df1,df2])`

Input:

- name - A single character specifying which distribution to use. The following choices are available:
 - Z or z Use the standard normal distribution.
 - T or t Use the Student- t distribution with `df1` degrees of freedom.
 - C or c Use the χ^2 distribution with `df1` degrees of freedom.
 - F or f Use the F distribution with `df1` and `df2` degrees of freedom.
- iopt - An integer specifying which function is to be evaluated. The following choices are available:
 - 1 Evaluate probability density function.
 - 2 Evaluate cumulative distribution function.
 - 3 Evaluate quantile function.
- n - Integer containing the number of points at which to evaluate the function ($n > 0$).

- x** - An array (if $n > 1$) of length n containing the values at which to evaluate the specified function, or a real scalar (if $n=1$) containing the single value at which to evaluate the function.
- a,b** - (Only in first form) Real scalars specifying at what points the function is to be evaluated. The n equally spaced points between a and b inclusive are the points to be evaluated.
- df1** - An integer containing the degrees of freedom for the t or χ^2 or the numerator degrees of freedom for the F (not needed if doing the standard normal distribution).
- df2** - An integer containing the denominator degrees of freedom for the F distribution (not needed if not doing the F distribution).

Output:

- X** - If $n=1$, a real scalar variable containing the desired value. If $n > 1$, an array of length n containing the desired values at the specified points.

Remark:

1. If the quantile function is desired, the values at which the function is to be evaluated must be in the open interval $(0,1)$.

DIVSDS Command

Purpose:

Calculate and/or divide and/or multiply seasonal standard deviations for a data set.

Format:

$Y = \text{DIVSDS}(X, n, d, \text{SDS}[, \text{iopt}])$

Input:

- X** - Array of length n containing data to be operated upon.
- n** - Integer containing the length of X (>0).
- d** - Integer containing the period of the seasons (>0).

- iopt** - Optional argument which if present is an integer indicating what operation is to be performed. The following options are available:
- 0 Calculate seasonal standard deviations but don't divide the data by them.
 - 1 Calculate and divide seasonal standard deviations.
 - 2 Multiply data by inputted SDS (this allows the inverse operation of **iopt**=1).
- SDS** - (Only if **iopt**=2) Array of length **d** containing values to multiply data by.

Output:

- Y** - (Unless **iopt**=0) Array of length **n** containing the result of dividing the input data by the seasonal standard deviations.
- SDS** - (Unless **iopt**=2) Array of length **d** (if **d**=1) or a real scalar variable (if **d**=1) containing seasonal standard deviations.

Remarks:

1. The argument **iopt** not being present is equivalent to **iopt**=1.
2. **Workspace**=**n**+**d**.

DOS Command

Purpose:

Allow the user to issue any command that can be issued at the DOS prompt.

Format:

DOS

Remarks:

1. The user's machine must have enough RAM to hold both **TIMESLAB** and **COMMAND.COM** and whatever command is issued. Further, a copy of **COMMAND.COM** must be on whatever device the user boots from. If there is not enough RAM or **TIMESLAB** can't find **COMMAND.COM**, **TIMESLAB** issues an appropriate message and reissues the prompt.

2. The DOS command and then DOS commands can be issued from within macros. Make sure to issue the EXIT command in the macro to return to TIMESLAB.

DOT Command

Purpose:

Find the inner product of the first n elements of two arrays.

Format:

`c=DOT(X,Y,n)`

Input:

- `X` - Array of length at least n .
- `Y` - Array of length at least n .
- `n` - Integer containing the number of elements of `X` and `Y` to use in the inner product.

Output:

- `c` - A real scalar variable containing the inner product of the first n elements of `X` and `Y`.

Remark:

1. `X` and `Y` can be the same array.

DOUBLE Command

Purpose:

Extend spectral arrays by symmetry about frequency .5.

Format:

`DOUBLE(fr,fi,Q)`

Input:

- `fr,fi` - Arrays of length $m = [Q/2] + 1$ to be extended to length Q .

Q - Integer containing the desired length of the extended arrays.

Output:

fr,fi - Arrays of length **Q** containing the extended arrays.

Remarks:

1. The extended values are given by

$$fr(Q-i) = fr(i+2) \quad \text{and} \quad fi(Q-i) = -fi(i+2)$$

for $i = 0, \dots, [Q/2] - 1$.

2. The value of $[Q/2]+1$ must equal the length of **fr** and **fi**.

3. **Workspace**=2**Q**.

DTAR Command

Purpose:

Fit autoregressive models to univariate data.

Format:

alpha=DTAR(**X**,**n**,**M**,**iopto**,**ioptl**,**ioptm**,**p**,**R0**,**rvar** [,**CAT**])

Input:

- X** - Array of length **n** containing data set.
- n** - Integer containing the length of the data set.
- M** - Integer containing either the AR order to use (if **iopto**=1) or the maximum AR order to use (if **iopto**=2).
- iopto** - Integer specifying whether to use order **M** (**iopto**=1) or to use the Parzen CAT criterion to determine an order less than or equal to **M** (**iopto**=2).
- ioptl** - Integer specifying whether or not to display on the screen a table of partial autocorrelations, “biased” and “unbiased” standardized residual variances, and the CAT criterion for orders 1 through **M** (1 means yes, 2 means no).
- ioptm** - Integer specifying which method is to be used in doing the estimation. The following choices are available:

- 1 Use the algorithm in Theorem 3.4.5 to find Yule-Walker estimates (YWE).
- 2 Use the Burg algorithm.
- 3 Each of the M partial autocorrelations from the algorithm in `ioptm=1` will be tested for significance (versus $\pm 2/\sqrt{n}$) and the parameters of the fitted model are found from the partials judged to be significantly different from zero.
- 4 This option is the same as 3 except the Burg algorithm is used.
- 5 This option uses an AIC-based method for testing for significance the partial autocorrelations from the YWE algorithm. The method starts with no partials in the subset. At each successive step, the partial largest in magnitude of those not in the subset is added and AIC is calculated for the new subset. If this AIC is smaller than the previous one the method continues. Otherwise the last added partial is dropped and the method stops. From the determined subset of partials, an AR order, coefficients, and error variance are determined.
- 6 This option is the same as 5 except that the Burg algorithm is used to get the partials.

Output:

- | | |
|--------------------|--|
| <code>p</code> | - Integer variable containing the order of the chosen model. |
| <code>R0</code> | - Real scalar variable containing the sample variance of the data. |
| <code>rvar</code> | - Real scalar variable containing the error variance of the chosen model. |
| <code>CAT</code> | - If this argument is included, it is an array of length M containing the values of the CAT criterion for orders 1 through M . |
| <code>alpha</code> | - Array of length p containing the coefficients of the chosen model. |

Remarks:

1. If $p=0$, then `rvar` and `alpha` are not formed.
2. $\text{Workspace}=2(n+M)+3M$.

DTARMA Command

Purpose:

Find exact maximum likelihood estimators of the parameters of an ARMA (or AR or MA) process.

Format:

DTARMA(X,n,p,q,maxit,eps,alpha,beta,rvar,m2l1,ier,del)

Input:

- X - Array of length n containing data to be used in the estimation procedure.
- n - Integer containing the length of the data.
- p - Integer containing the AR order to use.
- q - Integer containing the MA order to use.
- maxit - Integer ($1 \leq \text{maxit} \leq 500$) containing the maximum number of iterations in the optimization procedure (see Remark 1).
- eps - Real scalar containing the convergence criterion (see Remark 1).
- alpha - Array of length p containing starting values for the AR coefficients (see Remark 2).
- beta - Array of length q containing starting values for the MA coefficients (see Remark 2).

Output:

- alpha - Array of length p containing the values of the AR coefficients when DTARMA finishes (see Remark 3).
- beta - Array of length q containing the values of the MA coefficients when DTARMA finishes (see Remark 3).
- rvar - Real scalar variable containing the value of the error variance when DTARMA finishes (see Remark 3).
- m2l1 - Real scalar variable containing the value of -2 times the log likelihood evaluated at the output values of the parameters. This is useful in calculating the AIC.

- ier** - Integer variable indicating the presence of errors and/or convergence. If **ier** is 0, then there were no errors encountered and convergence was reached. If **ier** is 1, no errors were encountered but convergence was not reached. If **ier** is greater than 1, there was an error. This is usually because the model being estimated is very ill-conditioned (zeros of the AR or MA polynomials near the unit circle). Sometimes using different starting values will fix the problem. If nothing seems to help, consistent estimates can be obtained using the **ARMASEL** command.
- del** - A real scalar variable containing the sample variance of the values of the likelihood function at the vertices of the final simplex. This gives a measure of how close to convergence the algorithm has come. If this is less than **eps**, then **DTARMA** has come close to converging but not to a relative maxima.

Remarks:

1. The orders **p** and **q** can't both be zero.
2. If **p** or **q** is zero, then the corresponding coefficient arrays are ignored although an argument must be included in the argument list.
3. If convergence has been reached, it is to relative maximum of the likelihood. To be safe, it is good to try various starting values to see if they all converge to the same place. If convergence has not been reached, **DTARMA** can be called again with the output coefficients as new starting values.
4. The Nelder-Mead simplex derivative-free optimization method is used with the Kalman Filter Algorithm to evaluate the likelihood.
5. Starting values can be obtained by **CORRAR**, **CORRMA**, or **CORRARMA** for AR, MA, and ARMA processes respectively. **DTARMA** will check the input coefficients for stability. Both **alpha** and **beta** must give characteristic polynomials having all zeros outside the unit circle. **DTARMA** checks for this. The output **alpha** and **beta** are guaranteed to be stable. Sometimes just using zeros for starting values will work. Starting values can also be obtained from **ARMASEL** or **SEASEST**.
6. $\text{Workspace} = 3(p+q) + 2n + 5(\max(p, q+1) + 1) + (p+q)(p+q+1).$

DTFORE Command

Purpose:

Perform iterated AR forecasting of data (i.e., use a one lag, possibly non-stationary subset AR followed by a stationary full AR).

Format:

DTFORE(X,n,M1,M2,npr,lag,p,alpha1,alpha2,XP)

Input:

- X - Array of length n containing data.
- n - Integer containing the length of the data (>0).
- M1 - Integer indicating what the first AR model should be. Values of M1 have the following meanings:
 - j >0 The lag to use is the one less than or equal to M1 having largest least squares regression coefficient in absolute value.
 - j <0 The lag to use is $-j$.
 - 0 The first AR is not done.
- M2 - Integer indicating what the second AR should be. The values of M2 have the following meanings:
 - j >0 The order used is the one less than or equal to j as chosen by the CAT criterion.
 - j <0 The order used is $-j$.
 - 0 The second AR is not done.
- npr - Integer indicating how far into the future to predict (≥ 0).

Output:

- lag - Integer variable containing lag used for first AR.
- p - Integer variable containing order used for second AR.
- alpha1 - Real scalar variable containing the coefficient for the first AR.
- alpha2 - Array of length p containing coefficients for second AR.

- XP - Array of length $n+npr$. The first $lag+p$ elements are the first $lag+p$ elements of X , the next $n-lag-p$ elements are the one step ahead predictors of $X(lag+p+1), \dots, X(n)$. The last npr elements of XP (if $npr > 0$) are the 1 through npr step ahead predictors of the next npr values of the process.

Remark:

1. $Workspace = 4n + 2|M1| + 5|M2| + 2npr$.

ECHO Command

Purpose:

To provide a command that does the same thing as striking the Ctrl-PrtSc keystroke, that is, to toggle the printer-echo feature of a PC.

Format:

ECHO

Remarks:

1. After ECHO is issued the first time, everything that appears on the screen in the text mode (except the list of commands given in response to the HELP command) also appears on the printer. This is called entering the printer-echo mode. Issuing ECHO again turns off the printer-echo.
2. The PRINTSEL command must be issued first to tell TIMESLAB what printer is being used.

EDIT Command

Purpose:

Invoke the TIMESLAB full screen text editor.

Format:

1. EDIT(oldfn,newfn)
2. EDIT(fname)

Input:

- oldfn** - The name of the file to be edited. If there is no file of this name, EDIT assumes the user wants to create a new file having name **newfn**.
- newfn** - The name of the file to receive either the modified file or the newly created file.
- fname** - If the user wants the old and new files to be the same, then the second form of the command can be used.

Remarks:

1. The file names can have at most 15 characters.
2. See Section B.6 for more information about the EDIT command.

EIG Command

Purpose:

Calculate the eigenvalues and (optionally) eigenvectors of a real symmetric matrix.

Format:

```
evals=EIG(A,n,maxit,ier[,evecs])
```

Input:

- A** - An array of length $n*n$ containing the matrix for which to find eigenvalues.
- n** - Integer (≤ 56) containing the size of the matrix.
- maxit** - Integer (≤ 500) containing the maximum number of iterations to allow in the usual tridiagonalization and *Q-R* eigenvalue algorithm (usually 100 iterations is more than enough).

Output:

- ier** - Integer indicating whether convergence has been reached (0 means it has, 1 means it has not).
- evals** - Array of length **n** containing the eigenvalues in decreasing order.

- evecs** - If this argument is present, it is an array of length $n*n$ containing the eigenvectors of the matrix. The first n elements consist of the eigenvector corresponding to the first element of **evals**, the next n are the eigenvector for the second element of **evals**, and so on.

Remark:

1. $\text{Workspace}=3n^2+5n$.

END Command

Purpose:

Provide a terminator for a WHILE...END construct in a macro.

Format:

END

Remark:

1. See Section B.6 for more information on the looping constructs available in TIMESLAB macros.

ENDIF Command

Purpose:

Provide a terminator for an IF...ENDIF construct in a macro.

Format:

ENDIF

Remark:

1. See Section B.6 for more information on the looping constructs available in TIMESLAB macros.

ERASE Command

Purpose:

Erase a part of the screen in the graphics mode.

Format:

1. ERASE(cloc,rloc,ncs,nrs)
2. ERASE(kode)

Remarks:

1. In the first form, a rectangle is erased that is `ncs` columns by `nrs` rows in size and whose lower left-hand corner is located at the pixel in column `cloc` and row `rloc`.
2. In the second form, certain standard parts of the screen can be erased as specified by the following values of `kode`:
 - 0 The whole screen.
 - 1 The upper left quarter of the screen.
 - 2 The lower left quarter of the screen.
 - 3 The upper right quarter of the screen.
 - 4 The lower right quarter of the screen.
 - 5 The left half of the screen.
 - 6 The right half of the screen.
 - 7 The top half of the screen.
 - 8 The bottom half of the screen.
3. The screen has 640 columns and 200 rows of pixels. The columns are numbered from 0 to 639 from left to right, while the rows are numbered from 0 to 199 from bottom to top.

EXP Command

Purpose:

Calculate the exponential function.

Format:

1. $y = \text{EXP}(x)$
2. $Y = \text{EXP}(X, n)$

Remarks:

1. In the first form, the input x and the output y are real scalars.
2. In the second form, X is an input array of length at least n , and Y is an output array of length n .
3. The largest value that can be exponentiated is 87.5.

EXTEND Command

Purpose:

Extend a data set by the inverse operation of differencing.

Format:

$Y = \text{EXTEND}(X, n, \text{next}, d1[, d2])$

Input:

- | | | |
|---------------|---|--|
| X | - | Array of length n containing data. |
| n | - | Integer containing the length of X ($\geq d1 + d2$). |
| next | - | Integer containing the number of points to extend X (> 0). |
| $d1, d2$ | - | Integers containing differences. |

Output:

- | | | |
|-----|---|---|
| Y | - | Array of length $n + \text{next}$ containing X and the extended values. |
|-----|---|---|

Remarks:

1. The first n values of Y are the same as the first n values of X , while the next next values of Y are then found as described in Remarks 2 and 3.
2. If $d2$ is zero, then $Y(n+i) = Y(n+i-d1) + \text{ybar}$, where ybar is the average of the $d1$ differences of the first n X 's.
3. If $d2 \neq 0$, then $Y(n+i) = Y(n+i-d1) + Y(n+i-d2) - Y(n+i-d1-d2) + \text{zbar}$, where zbar is the sample mean of the data resulting from taking the $d1$ and $d2$ differences of the array X .

4. `Workspace=n+next.`

EXTRACT Command

Purpose:

Form an array by extracting elements of another array.

Format:

1. `Y=EXTRACT(X,n1,n2)` `Y` consists of `X(n1)` through `X(n2)`.
2. `Y=EXTRACT(X,n1,n2,inc,ny)` The output array `Y` consists of every `inc`th element of `X(n1)` through `X(n2)`. `ny` is the resulting number of elements in `Y`.
3. `Y=EXTRACT(X,XI,ni)` The output array `Y` consists of the elements of `X` having indices contained in the array `XI` of length `ni`.
4. `Y=EXTRACT(X,n,r,relation,ny)` The output array `Y` consists of the indices of the elements in the array `X` that satisfy a certain relation with the real scalar `r`. The `relation` argument can be one of `lt`, `le`, `eq`, `ge`, or `gt`. `ny` is an integer variable containing the resulting number of elements in `Y`.
5. `B=EXTRACT(A,ra,r1,c1,nr,nc)` The output array `B` is the `nr` by `nc` matrix consisting of the block of the input matrix `A` (which has `ra` rows) starting with its element in the `r1` row and `c1` column.
6. `B=EXTRACT(A,ra,r1,r2,ic,nc)` This form is the same as form 5 except that the `nc` columns of `A` having indices in the input array `ic` are extracted starting at row `r1`.

Remark:

1. For the first form, if only one point is extracted, then `Y` is a real scalar, otherwise it is an array. For the other forms, `Y` is always an array.

FFT Command

Purpose:

Calculate the discrete Fourier transform (DFT) or inverse discrete Fourier transform (IDFT) of complex arrays.

Format:

1. FFT(XR,XI,Q,n,sign,ZR,ZI)
2. FFT(X,Q,n,sign,ZR,ZI)

Input:

- | | | |
|-------|---|--|
| XR,XI | - | Arrays of length Q containing the real and imaginary parts of the complex numbers to be transformed. |
| X | - | Array of length Q containing the data set to be transformed if it is real. |
| Q | - | Integer containing the length of the input array(s). |
| n | - | Integer ($\leq Q$) containing the number of transformed values to store. |
| sign | - | Integer indicating whether the DFT or IDFT should be done (1 and -1 respectively). |

Output:

- | | | |
|-------|---|--|
| ZR,ZI | - | Arrays of length n containing the first n elements of the real and imaginary parts of the transform. |
|-------|---|--|

Remarks:

1. The second form is the same as the first except that the imaginary part of the numbers to be transformed is assumed to be zero.
2. The FFT command will use the FFT algorithm if the largest prime factor of Q is ≤ 23 and the product of the square-free prime factors is ≤ 210 . Otherwise, FFT will use the defining formula for the calculation. If this happens, the time required to calculate the transform may be quite extensive. For example, on a PC with a 4.77-MHz clock, it takes approximately 2.5 seconds to do a transform of length 101, about 10 seconds for Q=222, and 40 seconds for Q=444.
3. Workspace=2Q if the fast Fourier transform can be used, and 2Q+2n if the defining formula is used.

FILT Command

Purpose:

Provide a linear filter of an array, that is,

$$Y(i) = \sum_{j=0}^p \beta_j X(p+i-j), \quad i = 1, \dots, n-p.$$

Format:

`Y=FILT(X,beta,beta0,n,p)`

Input:

- `X` - Array of length `n` containing the data to be filtered.
- `n` - Integer containing the length of the data to be filtered.
- `beta` - Array of length `p` containing coefficients of lags $1, \dots, p$.
- `beta0` - Real scalar containing the coefficient for lag 0.
- `p` - Integer ($< n$) containing the length of the filter.

Output:

- `Y` - Array of length `n-p` containing the result of the filter.

Remark:

1. `Workspace=2n`.

GOTO Command

Purpose:

Provide unconditional branch for macros.

Format:

`GOTO(label)`

Input:

- label - The next line to be executed in the macro currently being executed is the line that has a ; in column 1 followed by the character string matching what is given by the argument of the GOTO command.

GRMENU Command

Purpose:

To place a graphics menu onto a graphics screen.

Format:

GRMENU(hmin,hmax,vmin,vmax)

Remarks:

1. Issuing this command places the graphics menu on a graphics screen from which one can print or save the screen, or use the FIND utility.
2. The arguments are the real-world values (as opposed to pixel coordinates) of the ends of the horizontal and vertical axes of a graph that is on the screen. Thus hmin must be less than hmax and vmin must be less than vmax.
3. If there is more than one set of axes on a screen, the user can successively use the FIND utility on the different axes by (1) issuing the PLOTSIZE command to tell TIMESLAB which axes to look at, (2) issuing the GRMENU command with the endpoints of that set of axes, and (3) striking F1 to invoke the FIND utility.

GS Command

Purpose:

To find the QR decomposition of an $(n \times m)$ matrix using the modified Gram-Schmidt decomposition algorithm.

Format:

GS(X,n,m,Q,R,ier)

Input:

- X** - Array of length $n*m$ containing the matrix to be decomposed.
- n,m** - Integers containing the dimensions of **X**.

Output:

- Q** - An array of length $n*m$ containing the orthogonal matrix in the decomposition.
- R** - An array of length $m*m$ containing the unit upper triangular matrix in the decomposition.
- ier** - Integer variable indicating whether or not the matrix to be decomposed was judged to be singular (0 means no, 1 means yes).

Remarks:

1. The arrays **Q** and **R** are not formed if **ier**=1.
2. $\text{Workspace} = nm + m^2$.

HELP Command

Purpose:

Provide an on-line manual.

Format:

1. **HELP**
2. **HELP(name)**
3. **HELP(name,d)**

Remarks:

1. The first form places a list of help topics on the screen.
2. The second form places a description of the topic having name **name** on the screen.
3. The third form is the same as the second except that the second argument is the letter of the device containing the file **TSLABHLP** (this must be A, B, or C).
4. The **HELP** command is not allowed in the graphics mode.

HIST Command

Purpose:

Calculate and plot the histogram of a data set.

Format:

1. HIST(X)
2. HIST(X,n)
3. HIST(X,n,nbins)
4. HIST(X,n,nbins,xmin,xmax,ymax)

Input:

- | | | |
|-------|---|---|
| X | - | Array of length n containing the data. |
| n | - | Integer ($\neq 0$) whose absolute value is the number of points of the array X to use in finding the histogram. If n is positive (negative), then the bars will be solid (not solid). |
| nbins | - | Integer ($0 < \text{nbins} \leq 200$) containing the number of intervals to use in the third and fourth forms. |
| xmin | - | Real scalar containing the value to place at the left end of the horizontal axis in the fourth form. |
| xmax | - | Real scalar containing the value to place at the right end of the horizontal axis in the fourth form. |
| ymax | - | Real scalar containing the value to place at the top of the vertical axis in the fourth form. |

Remarks:

1. In the first two forms, HIST uses

$$\text{nbins} = \left\lceil \frac{r_X}{3.5s_X} n^{1/3} \right\rceil + 1$$

as the number of intervals, where r_X and s_X are the range and standard deviation of the data respectively.

2. If ymax is entered as 0., hist will determine the scale on the vertical axis.

IF Command

Purpose:

Provide conditional and unconditional branches in a macro. See Section B.7 for examples of these branches.

Format:

1. `IF(v1.exp.v2)`
2. `IF(v1.exp.v2,label)`
3. `IF(var,nn,nz,np)`

Remarks:

1. In the first and second forms, `v1` and `v2` are two scalar variables to be compared, while `exp` is one of `lt`, `le`, `eq`, `ne`, `ge`, or `gt` for less than, less than or equal, equal, not equal, greater than or equal, or greater than respectively.
2. In the first form, if `v1` and `v2` are related by the specified expression, then control of the macro is passed to the next line of the macro, while if they are not so related, then control passes to the line after an `ENDIF` command that matches the `IF`.
3. In the second form, if `v1` and `v2` are related by the specified expression, then control passes to the line in the macro that begins with a semicolon followed by a string of characters matched by what is given by the second argument of the `IF`. Otherwise, control passes to the line following the `IF`.
4. In the third form, control of the macro is passed to the line having offset `nn`, `nz`, or `np` from the current line depending on whether the (integer or real) variable `var` is negative, zero, or positive. Blank lines and nondisplayed comment lines are not counted when determining offsets. A positive (negative) offset means that control will pass downward (upward) in the file.

INFO Command

Purpose:

Display a list of all variables that have been created in the current session.

Format:

INFO

Remarks:

1. The number of and names of all arrays, reals, integers, and characters are given.
2. The length and label of each array are given.
3. The value of each real, integer, and character is given.
4. The number of free elements out of the original 10,000 allowed for arrays is given.

INFQNT Command

Purpose:

Plot the informative quantile function for a data set.

Format:

1. INFQNT(X)
2. INFQNT(X,n)
3. INFQNT(X,n,xmed,xiqr)
4. INFQNT(X,n,xmed,xiqr,arg5)

Input:

- | | | |
|------|---|---|
| X | - | Array of length n containing the data. |
| n | - | Integer containing the length of the data set. |
| arg5 | - | Having a fifth argument means that the median and interquartile range are returned without the plot being formed. |

Output:

- | | | |
|------|---|--|
| xmed | - | Real scalar variable containing the median of the data set. |
| xiqr | - | Real scalar variable containing the interquartile range of the data set. |

Remarks:

1. An informative quantile plot is a plot of the sorted values of an array (after subtracting the median and dividing by twice the interquartile range) versus $(i - .5)/n$, $i = 1, \dots, n$.
2. The true informative quantile plot of a uniform random variable is a straight line. This line is always put on the plot.

INVPOLY Command

Purpose:

Find the first q coefficients of the reciprocal of the polynomial $g(z) = 1 + \sum_{j=1}^p \alpha_j z^j$.

Format:

`beta=INVPOLY(alpha,p,q)`

Input:

- `alpha` - Array of length `p` containing the coefficients of the polynomial to be inverted.
- `p` - Integer containing the length of `alpha` (>0).
- `q` - Integer containing the number of coefficients of the reciprocal to find (>0).

Output:

- `beta` - Array of length `q` containing first `q` coefficients of the reciprocal polynomial.

Remark:

1. `Workspace=q`.

LABEL Command

Purpose:

Allow the user to change the label for an array or to place a string of characters on a graphics screen.

Format:

1. LABEL(X)='some character string'
2. LABEL(X)=<Y>
3. LABEL(X)=<Y,Z>
4. LABEL(col,row,iopth)='some character string'
5. LABEL(name,offset)='some character string'

Remarks:

1. The first form gives the array X the label given by what is between the apostrophes. There can be at most 40 characters.
2. The second form replaces the current label for X with that of the array Y.
3. The third form takes the first 20 characters of each of the labels of the arrays Y and Z and forms a new label for X by concatenating them.
4. In the fourth form, the string of characters given between the apostrophes is placed on the screen starting at pixel location (col,row), where the lower left corner of the screen is location (0,0). The string is placed horizontally if iopth is 1 and vertically if iopth is 0. If placed horizontally, the lower left corner of the first character of the string is at (col,row), while if placed vertically, the upper left corner of the first character is at (col,row).
5. In the fifth form, the string of characters given between the apostrophes is placed in a position relative to the set of axes specified by the most recent call to the PLOTSIZE command (or to the default axes if PLOTSIZE has not been called) according to the values given for the two arguments. The first argument can be x, y, or cap (in upper or lowercase) which mean that the string is placed relative to the horizontal axis, the vertical axis, or the caption position respectively. The second argument is an integer specifying how far from the closest possible position the string is to be placed. An offset of 0 means that the string is placed as close as possible (a caption would be overwritten), while an offset of 1 means the string will be placed eight pixels away from the closest position. This would be down, left, and up for x, y, and cap respectively. The offset can also be negative which puts the string in the opposite direction. Strings placed on the horizontal or vertical axis are centered on the axis, while those placed at the caption are placed in line with captions generated by graphics commands.

6. In the first, fourth, and fifth forms, variable integers or reals can be inserted into a label by placing them within #’s (integers) or @’s (reals), for example,

```
LABEL(X)='AR(#nord#) Spectra, RVAR=@rvar@'
```

is the same as

```
LABEL(X)='AR(5) Spectra, RVAR=2.95'
```

if `nord=5`, `rvar=2.95`.

LENGTH Command

Purpose:

Find the number of elements in an array.

Format:

1. `LENGTH(X)`
2. `LENGTH(X,n)`

Remark:

1. The first form merely shows the length of `X` on the screen while the second form also places the length in the integer variable `n`.

LINE Command

Purpose:

Form an array consisting of equally spaced real numbers.

Format:

1. `X=LINE(n,a,b)`
2. `X=LINE(n,a,b,arg4)`

Remarks:

1. The first form finds $X(i)=a+bi$ for $i=1,\dots,n$.

2. In the second form, the fourth argument is used only as a signal that the second form is to be used and X is an array containing the n equally spaced reals between a and b inclusive.
3. This command is very useful in forming arrays of constants (first form with $b=0$), indices of time series (second form with a being the first time minus the sampling interval and b being the sampling interval), or finding the values at which to evaluate a distribution or polynomial (e.g., to get $-1, -.99, \dots, .99, 1$, one can use the second form with $n=201$, $a=-1$, and $b=1$).
4. `Workspace=n`.

LIST Command

Purpose:

Display the values of one or more variables or direct the display to a file (including the printer).

Format:

1. `LIST(list of scalar variables)`
2. `LIST(X)`
3. `LIST(X,0)`
4. `LIST(X,n)`
5. `LIST(X,n,ng,FORMAT)`

Remarks:

1. The first form displays the values of the (real, integer, or character) variables in the argument list.
2. The second form lists all of the elements of the array X , five per line in 5G14.6 format.
3. The third form lists only the label of the array X .
4. The fourth form lists the first n elements of the array X in 5G14.6 format.
5. The fifth form lists the first n elements of the array X , ng elements per line, in the format specified by `FORMAT`, which is a Fortran format statement without its parentheses, for example, `LIST(X,100,10,10F7.2)`.

6. In any of the above forms, putting another argument which is a file name between apostrophes sends the display to that file instead of to the screen. This includes the printer (which has file name `prn` or `lpt1` in either upper or lowercase).
7. The format `5G14.6` means that there are five numbers per line, with six digits used for each number unless the number is less than `.01` or greater than `106`, in which case scientific notation is used.
8. When listing an array, the index of the first element of each row in the display is listed also.
9. Only one array can be displayed per command and scalars and arrays cannot be listed in the same command.

LISTM Command

Purpose:

Display a matrix or three-dimensional array.

Format:

`LISTM(A,n,m[,k])`

Input:

- | | |
|----------|--|
| A | - Array to be listed. If considered a matrix, then its first <code>n</code> elements are its first column, and so on. If considered a three-dimensional array, its first <code>n</code> elements are the first column of the first matrix, the next <code>n</code> are the second column, and so on. |
| n | - Integer containing the number of rows of the matrix(ces). |
| m | - Integer containing the number of columns of the matrix(ces). |
| k | - Integer containing the number of matrices. |

Remark:

1. The display is done in `6F12.6` format.

LISTSP Command

Purpose:

Display the elements of a spectral array along with their frequencies.

Format:

LISTSP(f,Q)

Input:

- f - Array of length $m = [Q/2] + 1$ containing some spectral quantity at the frequencies $(j - 1)/Q$, $j = 1, \dots, m$.
- Q - Integer containing the number of frequencies at which the spectral quantity was calculated.

LOGE Command

Purpose:

Calculate the natural log of a real scalar or of an array.

Format:

1. $y = \text{LOGE}(x)$ The input x and the output y are real scalars.
2. $Y = \text{LOGE}(X, n)$ The input X is an array of length at least n , and the output Y is an array of length n containing the natural log of the elements of X .

Remarks:

1. If any of the input reals are nonpositive, an error message is displayed and the output is not found.
2. Workspace=M.

MACORR Command

Purpose:

Find the variance $R(0)$ and correlations $\rho(1), \dots, \rho(M)$ of an $\text{MA}(q, \boldsymbol{\beta}, \sigma^2)$ process.

Format:

`rho=MACORR(beta,q,rvar,M,R0)`

Input:

- `beta` - Array of length `q` containing coefficients $\boldsymbol{\beta}$.
- `q` - Integer containing order q (>0).
- `rvar` - Real scalar containing error variance σ^2 (>0).
- `M` - Integer containing the number of autocorrelations to find (>0).

Output:

- `R0` - Real scalar variable containing the variance of the MA process.
- `rho` - Array of length `M` containing the MA autocorrelations.

Remark:

1. `Workspace=M`.

MACRO Command

Purpose:

Invoke or reinvoke a macro.

Format:

1. `MACRO(fname)`
2. `MACRO(fname,label)`
3. `MACRO`

Remarks:

1. The first form invokes the macro in the file having name `fname`.
2. The second form invokes the macro in the file having name `fname` at the line in the file having (1) a semicolon in column 1, (2) then a string of characters matching that given as the second argument.
3. The third form reinvoles the most recently interrupted macro.
4. See Section B.7 for more information about macros.

MADT Command

Purpose:

Generate a realization from a Gaussian MA(q, β, σ^2) process.

Format:

`X=MADT(beta,q,rvar,seed,n)`

Input:

- | | |
|-------------------|--|
| <code>beta</code> | - Array of length <code>q</code> containing coefficients β . |
| <code>q</code> | - Integer containing order q (>0). |
| <code>rvar</code> | - Real scalar containing error variance σ^2 (>0). |
| <code>seed</code> | - Real scalar containing the seed for the random number generator. |
| <code>n</code> | - Integer ($>q$) containing the number of observations desired. |

Output:

- | | |
|----------------|--|
| <code>X</code> | - Array of length <code>n</code> containing the realization. |
|----------------|--|

Remark:

1. `Workspace=n`.

MASP Command

Purpose:

Calculate the spectral density of an MA(q, β, σ^2) process.

Format:

`f=MASP(beta,q,rvar,Q)`

Input:

- `beta` - Array of length `q` containing coefficients β .
- `q` - Integer containing order q (>0).
- `rvar` - Real scalar containing error variance σ^2 (>0).
- `Q` - Integer ($>q$) containing the number of frequencies between 0 and 1 at which to calculate the spectral density. The greatest prime factor of `Q` can be at most 23, and the product of its square-free prime factors can be at most 210.

Output:

- `f` - Array of length $m = [Q/2] + 1$ containing MA spectra at frequencies $(j - 1)/Q$, $j = 1, \dots, m$.

Remark:

1. `Workspace=2Q`.

MAXMIN Command

Purpose:

Find the values and indices of the maximum and minimum of an array.

Format:

`MAXMIN(X,n,xmax,imax,xmin,imin)`

Input:

- `X` - Array of length `n` for which to find max and min.
- `n` - Integer containing the length of `X`.

Output:

- `xmax` - Real scalar containing the largest value in the array `X`.
- `xmin` - Real scalar containing the smallest value in the array `X`.
- `imax` - Integer containing the index in `X` of `xmax`.

imin - Integer containing the index in **X** of **xmin**.

MCHOL Command

Purpose:

To find the modified Cholesky decomposition of a symmetric, positive definite matrix.

Format:

MCHOL(A,n,L,D,ier)

Input:

- A** - Array of length $n*n$ containing the matrix to be decomposed.
- n** - Integer containing the size of the square matrix to be decomposed.

Output:

- L** - Array of length $n*n$ containing the unit lower triangular factor in the decomposition.
- D** - Array of length $n*n$ containing the diagonal factor in the decomposition.
- ier** - Integer indicating whether or not **A** was judged to be positive definite (0 means positive definite, 1 means not positive definite).

Remarks:

1. If **ier**=1, then **L** and **D** are not formed.
2. $\text{Workspace}=2n^2+n$.

MDEL Command

Purpose:

To delete specified rows of a matrix.

Format:
$$B = \text{MDEL}(A, n, m, r1, \dots, rk)$$
Input:

- A - Array of length $n \times m$ containing the matrix whose rows are to be deleted.
- n, m - Integers containing the size of the matrix.
- r1, ... - Integers containing the numbers of the rows to be deleted.

Output:

- B - Array containing the $(n-k) \times m$ output matrix.

Remark:

1. $\text{Workspace} = (n-k)m$.

MINV Command

Purpose:

To find the inverse of a matrix.

Format:
$$B = \text{MINV}(A, n, ier)$$
Input:

- A - Array of length $n \times n$ containing the matrix to be inverted.
- n - Integer containing the dimension of A.

Output:

- ier - Integer indicating whether or not A was judged nonsingular (0 means nonsingular, 1 means singular).
- B - Array of length $n \times n$ containing the inverse of A.

Remarks:

1. If $ier=1$, then B is not formed.
2. $\text{Workspace} = 2n^2$.

MMULT Command

Purpose:

To multiply matrices.

Format:

1. $C=MMULT(A,B,n,m,k)$
2. $C=MMULT(A,B,n,m)$
3. $C=MMULT(A,B,n)$

Remarks:

1. In the first form, A is $n \times m$, B is $m \times k$, and $C=AB$.
2. In the second form, A is $n \times m$, B is $n \times m$, and $C=A'B$.
3. In the third form, A and B are $n \times n$ and $C=AB$.
4. The workspaces for the three forms are $nk+nm$, m^2 , and $2n^2$ respectively.

MULTPOLY Command

Purpose:

Find the coefficients of the product of the two polynomials

$$g(z) = 1 + \sum_{j=1}^p \alpha_j z^j \quad \text{and} \quad h(z) = 1 + \sum_{k=1}^q \beta_k z^k.$$

Format:

$gama=MULTPOLY(alpha,beta,p,q)$

Input:

- | | |
|--------------|---|
| alpha | - Array of length p containing the coefficients of the first polynomial. |
| beta | - Array of length q containing the coefficients of the second polynomial. |
| p | - Integer containing the degree of the first polynomial ($p > 0$). |

- q** - Integer containing the degree of the second polynomial ($q > 0$).

Output:

- gama** - Array of length $p+q$ containing the coefficients of product polynomial.

NOTES Command

Purpose:

To allow users to create and access their own help files. User-defined help files are called notes.

Format:

1. **NOTES(fname)**
2. **NOTES(fname,notename)**

Remarks:

1. A note consists of a series of lines in an ASCII file. The first line of a note consists of the name of the note starting in column 1. The rest of the note is a series of lines (column 1 must be blank in each line) containing information about something.
2. The first form results in a list of the note names on the file having name **fname**.
3. The second form results in the display of the note having name **notename** in the file called **fname**.

OVEROFF Command

Purpose:

Tell TIMESLAB to henceforth ask for permission to replace any previously defined variable whenever a command would result in such a replacement.

Format:

OVEROFF

OVERON Command

Purpose:

Tell TIMESLAB to henceforth automatically overwrite any previously defined variable whenever a command would result in such a replacement.

Format:

OVERON

Remarks:

1. The BATCHON command also causes automatic overwriting.
2. OVERON is cancelled by OVEROFF.
3. When TIMESLAB starts, it is in the OVERON mode.

PAGE Command

Purpose:

Position the printer to the top of the next page. On a nonlaser printer this is where the print head is on the paper when the printer is turned on or the PC is reset. On a laser printer, PAGE causes a page eject.

Format:

PAGE

PARCORR Command

Purpose:

Calculate partial autocorrelations and optionally residual variances for a data set.

Format:

part=PARCORR(X,n,M[,rvar])

Input:

- X** - Array of length n containing the data.
- n** - Integer containing the length of **X**.
- M** - Integer ($0 < M < n$) containing the number of partial auto-correlations to find.

Output:

- part** - Array of length M containing partial autocorrelations.
- rvar** - If this argument is present, it is an array of length M containing standardized residual variances.

Remark:

1. $\text{Workspace}=2n+4M$.

PARTAR Command

Purpose:

Calculate the coefficients of an $\text{AR}(p, \alpha, \sigma^2)$ process given its partial auto-correlations.

Format:

$\text{alpha}=\text{PARTAR}(\text{part}, p)$

Input:

- part** - Array of length p containing partial autocorrelations.
- p** - Integer containing order p (>0).

Output:

- alpha** - Array of length p containing coefficients α .

PAUSE Command

Purpose:

Provide a means of obtaining user input during execution of a macro or else just have a macro pause.

Format:

PAUSE

Remarks:

1. When TIMESLAB encounters a PAUSE command during execution of a macro it issues the message "Strike F9 to break MACRO, anything else to go on." Striking F9 at this point will cause the macro to stop executing and TIMESLAB will issue its usual prompt. Then the user can issue other commands and then restart the macro by entering the command MACRO.
2. The writer of the macro can precede the PAUSE command by some echoed comments, that is, comment lines that begin with two semicolons describing what variables need to be defined.

PLOT Command

Purpose:

General purpose Y versus X plot.

Format:

1. PLOT(X)
2. PLOT(X, n)
3. PLOT(X, n, x_{\min}, x_{\max})
4. PLOT($X, n, h_{\min}, h_{\max}, v_{\min}, v_{\max}$)
5. PLOT(X, Y, n)
6. PLOT($X, Y, n, x_{\min}, x_{\max}, y_{\min}, y_{\max}$)

Remarks:

1. The first form graphs the elements of the input array X versus their index with PLOT determining the scale of the plot so that the ends of the axes correspond to the minimum and maximum of X on the vertical axis, the number 1 on the left of the horizontal axis, and the number of points being plotted put at the right end. The entire array X is plotted.
2. The second form is the same as the first except that the first n elements of the array X are plotted.

3. The third form is the same as the second except that the user specifies `xmin` and `xmax`, that is, the values to be used at the ends of the vertical axis.
4. The fourth form again plots the first `n` elements of `X` versus their index except now the values at the ends of both axes are specified by the user in `hmin`, `hmax`, `vmin`, `vmax`.
5. In the fifth form, an array `Y` is plotted on the vertical axis, an array `X` is plotted on the horizontal axis, and the values at the ends of the axes are the actual minima and maxima of the arrays.
6. The sixth form is the same as the fifth except that the values at the ends of the axes are specified by the user in the real scalar input values `xmin`, `xmax`, `ymin`, `ymax`.
7. In any of the forms, if `n` is negative then a point plot is produced. This allows a scatterplot to be produced.

PLOT2 Command

Purpose:

Superimpose the graphs of two arrays.

Format:

1. `PLOT2(Y1,Y2,ny1,ny2,type1,type2)`
2. `PLOT2(Y1,Y2,ny1,ny2,type1,type2,X)`
3. `PLOT2(Y1,Y2,ny1,ny2,type1,type2,xmin,xmax,ymin,ymax)`
4. `PLOT2(Y1,Y2,ny1,ny2,type1,type2,X,xmin,xmax,ymin,ymax)`

Input:

- `Y1` - Array of length `ny1`.
- `Y2` - Array of length `ny2`.
- `ny1,ny2` - Integers containing the lengths of `Y1` and `Y2`.
- `type1` - Integer indicating how `Y1` is to be plotted. The following values are available:
 - 1 Line plot.
 - 2 Point plot.

- 3 If both `type1` and `type2` are 3, then an x is placed at the i th points of both plots and vertical lines connect them.
 - 4 Line plot with x's at each point.
- `type2` - Same as `type1` except for Y2.
- `X` - In the second and fourth forms, an array containing values for the horizontal axis.
- `xmin` - In the third and fourth forms, a real scalar containing the value to be used at the left end of the horizontal axis.
- `xmax` - In the third and fourth forms, a real scalar containing the value to be used at the right end of the horizontal axis.
- `ymin` - In the third and fourth forms, a real scalar containing the value to be used at the bottom of the vertical axis.
- `ymax` - In the third and fourth forms, a real scalar containing the value to be used at the top of the vertical axis.

Remark:

1. In the first and third forms, the values on the horizontal axis are the numbers 1, 2, ...

PLOT CSP Command

Purpose:

To plot a spectral array on a nonlog scale.

Format:

`PLOT CSP(f,Q)`

Remarks:

1. `f` is an array of length $[Q/2]+1$.
2. `f(i)` versus $(i-1)/Q$, $i=1,...,[Q/2]+1$, is plotted.

PLOTK Command

Purpose:

To allow the superimposing of any number of plots on the same axes.

Format:

```
PLOTK(X,Y,n,k,type[,xmin,xmax,ymin,ymax])
```

Input:

- k** - Integer containing the number of plots to be superimposed.
- X,Y** - Arrays containing the arrays to be plotted.
- n** - In general, an array of length k determining the number of points in each plot. The first n(1) elements of X and Y are plotted versus each other, the next n(2) elements are plotted versus each other, and so on. If each plot has the same number of points, then the common number can be inputted as an integer n.
- type** - Array of length k containing the type of plot to be produced for each plot. A value of 2 gives a line plot, 10, 11, 12, 13, and 14 give a line plot with symbols at each point (10 gives a single pixel, 11 gives an x, 12 gives a square, 13 gives a lower triangle, and 14 gives an upper triangle), while 30–34 give just point plots with 30 being a single pixel, 31 being an x, 32 a square, 33 a lower triangle, and 34 an upper triangle. If k is 1, then type can be a scalar.

Remarks:

1. If the last four arguments are included, they are real scalars specifying the endpoints of the axes.
2. The labels for Y and X are placed above the completed screen.

PLOTOFF Command

Purpose:

To switch the screen from the graphics mode to the text mode.

Format:

PLOTOFF

Remark:

1. If the screen is already in the text mode, then issuing this command will clear the screen.

PLOTON Command

Purpose:

To switch the screen from the text mode to the graphics mode.

Format:

PLOTON

Remarks:

1. After this command is issued, the prompt will stay at the top of the screen so that the user can issue a series of graphics commands without the prompt and output of commands interfering with what is on the rest of the screen.
2. If the screen is already in the graphics mode, then this command will clear the screen.

PLOTSIZE Command

Purpose:

To specify various parameters to be used in graphics commands that are issued in the future.

Format:

1. PLOTSIZE(npixx,npixy,nx1,ny1,nticsx,nticsy,nclabx,ndecx,nclaby,ndecy)
2. PLOTSIZE(npixx,npixy,nx1,ny1,nticsx,nticsy)
3. PLOTSIZE(npixx,npixy,nx1,ny1)
4. PLOTSIZE(kode)

Input:

- npixx** - The length in pixels of the horizontal axis.
- npixy** - The length in pixels of the vertical axis.
- nx1** - The column number where the origin is to be.
- ny1** - The row number where the origin is to be.
- nticsx** - The number of tic marks on the horizontal axis.
- nticsy** - The number of tic marks on the vertical axis.
- nclabx** - The number of characters to be allowed for the numerical labels on the tic marks on the horizontal axis. This includes a decimal point and a minus sign if there is one.
- ndecx** - The number of places to the right of the decimal point in the tic mark labels on the horizontal axis.
- nclaby** - The number of characters to be allowed for the numerical labels on the tic marks on the vertical axis. This includes a decimal point and a minus sign if there is one.
- ndecy** - The number of places to the right of the decimal point in the tic mark labels on the vertical axis.
- kode** - The fourth form is equivalent to the third form with the following values of **kode** using the indicated values of **npixx**, **npixy**, **nx1**, **ny1**:
 - 0 480 120 55 20 (default values)
 - 1 260 60 50 120 (upper left quarter of the screen)
 - 2 260 60 50 20 (lower left quarter of the screen)
 - 3 260 60 50 20 (upper right quarter of the screen)
 - 4 260 60 50 20 (lower right quarter of the screen)
 - 5 260 120 50 20 (left half of the screen)
 - 6 260 120 360 20 (right half of the screen)
 - 7 480 60 50 110 (upper half of the screen)
 - 8 480 60 50 20 (lower half of the screen)

Remarks:

1. When **TIMESLAB** is started, the default values of the arguments are:
npixx:480, **npixy**:120, **nx1**:55, **ny1**:30, **nticsx**:10, **nticsy**:10, **nclabx**:8, **ndecx**:2, **nclaby**:6, **ndecy**:2.

2. The value of $\text{npixx}+\text{nx1}+8*\text{nclaby}+5$ should be less than 640 for the horizontal axis to fit on the screen, while the value of $\text{npixy}+\text{ny1}+12$ should be less than 199 for the vertical axis to fit.
3. A value of zero for nclabx or nclaby results in there being no tic mark labels on the horizontal or vertical axis respectively.

PLOTSP Command

Purpose:

Plot a spectral array or several spectral arrays on a log scale.

Format:

1. `PLOTSP(f,Q,d)`
2. `PLOTSP(f1,Q,d1,...,fk,Q,dk)`

Remarks:

1. The first form produces a plot of

$$\log(f(i)/d) \text{ versus } (i-1)/Q,$$

for $i = 1, \dots, [Q/2] + 1$.

2. The second form superimposes several such plots. Note that Q must be the same for each plot while the divisors $d1, \dots, dk$ can be different. TIMESLAB knows how many plots to superimpose because there are three arguments for each one.
3. The label (or labels) for f (or $f1, \dots, fk$) are placed above the completed screen.

POLAR Command

Purpose:

Find amplitudes and phases of complex numbers.

Format:

1. `POLAR(xr,xi,amp,phase)`
2. `POLAR(xr,xi,n,amp,phase)`

Input:

- xr,xi** - In the first form, real scalars containing the real and imaginary parts of a complex number, and in the second form arrays of length at least **n** containing the real and imaginary parts of a set of complex numbers.
- n** - Integer containing the length of **xr** and **xi** (>0).

Output:

- amp** - Real scalar (first form) or an array (second form) of length **n** containing the amplitudes.
- phase** - Real scalar (first form) or an array of length **n** (form 2) containing the phases.

Remarks:

1. See Problem T1.2 for a discussion of how the amplitude and phase of a complex number are defined.
2. $\text{Workspace}=2n$ in the second form.

POLY Command

Purpose:

Evaluate a polynomial $\sum_{j=0}^p \alpha_j x^j$ at a specified set of values.

Format:

1. **Y=POLY(alpha,p,n,a,b)**
2. **Y=POLY(alpha,p,n,X)**

Input:

- alpha** - Array of length **p+1** containing the coefficients of the polynomial of degree **p** (**alpha(i)** is the coefficient of power **i-1**).
- p** - Integer containing the degree of the polynomial (>0).
- n** - Integer containing the number of points at which to evaluate the polynomial (>0).
- a,b** - In the first form, input scalars telling POLY to evaluate the polynomial at the **n** equally spaced values between **a** and **b** inclusive.

- X** - In the second form, an array of length n containing the points at which to evaluate the polynomial.

Output:

- Y** - If $n=1$, b is ignored and Y is a real scalar variable containing the desired value of the polynomial. If $n>1$, then Y is an array of length n containing the values of the polynomial.

POLYROOTS Command

Purpose:

To find the zeros of the polynomial

$$g(z) = \sum_{j=0}^p \alpha_j z^j.$$

Format:

POLYROOTS(alpha,alpha0,p,maxit,rr,ri,ier)

Input:

- alpha** - Array of length p containing α_1 through α_p .
alpha0 - Real scalar containing α_0 .
p - Integer containing the degree of the polynomial (>0).
maxit - Integer ($1 \leq \text{maxit} \leq 1000$) containing the maximum number of iterations in the procedure.

Output:

- rr** - Array of length p containing the real parts of the roots.
ri - Array of length p containing the imaginary parts of the roots.
ier - Integer variable indicating whether or not the procedure converged (0 means yes, 1 means no).

Remarks:

1. If $\text{ier}=1$, then rr and ri are not formed.
2. $\text{Workspace}=9(p+1)$.

PRINT Command

Purpose:

Print an array or send its display to a file.

Format:

The PRINT command is identical to the form of the LIST command that directs the output to the printer except that the argument specifying the file name is not used.

PRINTER Command

Purpose:

Set printer characteristics.

Format:

PRINTER(*code1*, ..., *codek*)

Remarks:

1. The arguments are all input integers.
2. If *code1* is not 27, then the arguments have the following meanings for an IBM Graphics Printer (or compatible):

Options in PRINTER Command

IOPT	Meaning	IOPT	Meaning
1	Compressed print	2	Compressed print off
3	Use character set 1	4	Use character set 2
5	Emphasized print on	6	Emphasized print off
7	Double strike on	8	Double strike off
9	Wide size print on	10	Wide size print off
30+n	Advance n lines		

3. If `iopt1=27`, then the rest of the arguments are sent directly to the printer. The user's printer manual should be consulted to find what codes are useful. An example is `PRINTER(27,27,38,108,49,79)` which puts the H-P Laserjet into the landscape mode.
4. If a printer other than the IBM Graphics Printer is being used, then the `PRINTSEL` command should be used to tell TIMESLAB what the printer is.

PRINTSEL Command

Purpose:

To select which of three possible types of printers is attached to the first parallel port.

Format:

`PRINTSEL(n)`

Input:

- `n` - Integer specifying which printer type is being used. The following choices are available:
- 1 IBM Graphics Printer (or compatible).
 - 2 HP Laserjet Printer (or compatible).
 - 3 Toshiba 321 (or compatible).

PROMPTOFF Command

Purpose:

Henceforth the lines of a macro are not displayed on the screen as they are issued.

Format:

PROMPTOFF

PROMPTON Command

Purpose:

Cancel the effect of PROMPTOFF.

Format:

PROMPTON

PSOFF Command

Purpose:

Cancel the effect of PSON.

Format:

PSOFF

PSON Command

Purpose:

Henceforth any plot that goes to the screen will also be dumped to the printer until PSOFF is issued.

Format:

PSON

Remark:

1. Screen dumps can be printed on either an IBM Graphics Printer (or compatible), an HP Laserjet Printer (or compatible), or a Toshiba 321 (or compatible). See the PRINTSEL command and Section B.5.4 for more information.

QTEST Command

Purpose:

Perform the modified portmanteau test for white noise.

Format:

`pval=QTEST(rho,M,p,q,n,Q)`

Input:

- `rho` - Array of length `M` containing autocorrelations of the data being tested.
- `M` - Integer containing the length of `rho`.
- `p,q` - If `rho` are autocorrelations from an ARMA(p,q) model, then `p` and `q` are integers containing the orders. Otherwise, they should be given the value zero. `M-p-q` must be positive.
- `n` - Integer ($> M$) containing the sample size for the data being tested.

Output:

- `Q` - Real scalar variable containing the test statistic.
- `pval` - Real scalar variable containing the p -value of the test.

QUIT Command

Purpose:

Quit execution of TIMESLAB and return to DOS.

Format:

`QUIT`

READ Command

Purpose:

Read a data set from a disk file.

Format:

```
READ(fname,X,n[,nskip])
```

Input:

- `fname` - Name of the file containing the data.
- `nskip` - If this argument is present, it is an integer containing the number of data sets to skip on the file before reading the set that is desired. If `nskip` is not present, then the first data set is read.

Output:

- `X` - Array of length `n` containing the data.
- `n` - Integer variable containing the length of `X`.

Remark:

1. See Section B.4.8 for more information about data files.

RECORD Command

Purpose:

Keep a record on a file of the output displayed by several of the TIMES-LAB commands.

Format:

1. RECORD(fname)
2. RECORD(close)

Remarks:

1. The first form begins the recording and tells TIMESLAB to use the file having name `fname`. The second form tells TIMESLAB to end the recording and to close the file. If a file already exists having name `fname`, RECORD will append the new material onto the end of the existing file. If the device being used for recording becomes full, RECORD will stop adding material to the file without necessarily telling the user.
2. Any command entered at the keyboard will be recorded as well as information displayed by the stepwise form of the REG command and information displayed by the ARMASEL, DTAR, INFO, LIST, LISTM, and SEASEST commands.

REG Command

Purpose:

Regress the array Y on the $(n \times m)$ matrix X .

Format:

1. REG($Y, X, n, m, \text{beta}, \text{RSS}, t, \text{res}$)
2. REG($Y, X, n, m, \text{pval}, \text{kopt}, k, \text{beta}, \text{ind}, \text{RSS}$)
3. REG($Y, X, n, m, \text{beta}, \text{VIBETA}, \text{RSS}, \text{ni}, t, \text{res}, \text{lrssio}$)

Remarks:

1. The first form is referred to as the simple form, the second as the stepwise form, and the third as the general form (see Section A.5).
2. The workspaces required are:

$$\left\{ \begin{array}{ll} 2(n+m) + nm + (m+1)^2 & \text{for form 1} \\ (m+1)n + (m+1)^2 + 3m & \text{for form 2} \\ (m+1)^2 + m + n & \text{for form 3 and } n \geq m \\ (n+m)^2 + m + n & \text{for form 3 and } n < m. \end{array} \right.$$

REPLACE Command

Purpose:

To replace selected elements of an array.

Format:

1. `Y=REPLACE(X,XI,R,nr)`
2. `Y=REPLACE(X,n,iopt,a[,b,c])`

Remarks:

1. In the first form, the output array `Y` is the same as the input array `X` except that the elements of `X` having indices contained in the array `XI` (which is of length `nr`) are replaced by the corresponding elements of the array `R`. Thus the element of `X` having index `XI[i]` is replaced by `R[i]`, for $i=1, \dots, nr$. If all of the replacing values are the same, then `R` can be entered as a real scalar.
2. The second form is used to do various kinds of truncations of the first `n` elements of the input array `X`. The following options are available as specified by the input integer `iopt` and how many arguments there are:
 - 1 Values of `X` that are less than `a` are replaced by `a` (4 arguments).
 - 2 Values of `X` that are greater than `a` are replaced by `a` (4 arguments).
 - 3 Values of `X` that are less than `a` are replaced by `a`, while values of `X` that are greater than `b` are replaced by `b` (5 arguments).
 - 4 Values of `X` between `a` and `b` (inclusive) are replaced by `c` (6 arguments).
 - 5 Values of `X` that are less than or equal to `a` are replaced by `b`, while values of `X` that are greater than `a` are replaced by `c` (6 arguments).

RESCREEN Command

Purpose:

To display a previously created graphics screen or screens.

Format:

1. RESCREEN(*fname*)
2. RESCREEN(*fn1*, ..., *fnk*, *iopt*)
3. RESCREEN(*fn*, *j*, *k*, *iopt*)

Remarks:

1. The first form recreates the screen contained in the file called *fname*.
2. The second form recreates successively screens in files *fn1*, ..., *fnk* either erasing in between (*iopt*=1) or superimposing them all (*iopt*=2).
3. The third form is similar to the second form except that file names all start with the characters specified by *fn* (which can be any string of characters) and end with the integers *j*, *j+1*, ..., *k*. Further, *iopt* must be either 3 or 4 which has the same meaning as 1 or 2 in the second form.
4. In the second and third forms, adding 10 to *iopt* causes a pause between screens until a carriage return is struck.

RESTART Command

Purpose:

To restart TIMESLAB.

Format:

RESTART

Remark:

1. When TIMESLAB sees this command it begins execution at its own first line. Thus the welcoming screen reappears and everything is reinitialized.

REVERSE Command

Purpose:

To create an array which is the same as another with its elements reversed.

Format:

`Y=REVERSE(X,n)`

Input:

- `X` - Array of length `n`.
- `n` - Integer containing length of `X`.

Output:

- `Y` - Array of length `n` containing the elements of `X` in reverse order.

ROOTSPOLY Command

Purpose:

To find the coefficients of a polynomial

$$g(z) = 1 + \sum_{j=1}^p \alpha_j z^j$$

given its zeros.

Format:

`ROOTSPOLY(rr,ri,p,alpha)`

Input:

- `rr,ri` - Arrays of length `p` containing the real and imaginary parts of the zeros of the polynomial.
- `p` - Integer containing the degree of polynomial.

Output:

- `alpha` - Array of length `p` containing coefficients.

Remarks:

1. In the POLYROOTS command, we allowed the coefficient of z^0 to be different than 1. Thus if the commands POLYROOTS and ROOTSPOLY are issued successively, the coefficients from ROOTSPOLY will be the original coefficients divided by the original constant term.
2. $\text{Workspace}=9(p+1)$.

SAVE Command

Purpose:

To save an array onto a disk file.

Format:

`SAVE(X,n,fname)`

Input:

- `X` - Array of length `n` to be saved.
- `n` - Integer containing the length of `X`.
- `fname` - Name of file where array will be saved.

Remarks:

1. The array is saved in 5E14.6 format.
2. If the disk file already exists, then this data set will be appended to the file.

SAVESC Command

Purpose:

To save a compressed image of the next graphics screen onto a file.

Format:

`SAVESC(fname)`

Remarks:

1. If this command is issued in the text mode, then the next time that a graph is formed using one of the HIST, INFQNT, PLOT, PLOTCSPP, PLOTK, PLOTSP, or PLOT2 commands, the screen will be saved onto a disk file whose name is given as the argument of the SAVESC command.
2. If this command is issued in the graphics mode, then whatever is on the screen (with the top eight rows of pixels erased) will be saved to the specified file.

SEASEST Command

Purpose:

Fit a multiplicative subset ARMA model to data X of length n .

Format:

SEASEST($X,n,ords,coeffs,lags,it,eps,nb,rvar,ier,sds[,e]$)

Input:

- | | |
|----------|--|
| X,n | - Data and length of data. |
| $ords$ | - An array of length 5 containing the full and subset AR orders, followed by the full and subset MA orders, followed by a 1 if a constant term is in the model or a 0 if it is not. |
| $lags$ | - An array containing the lags (if any) in the model. If both the subset AR and MA orders are zero, no array called $lags$ need be formed, but an argument must be included. |
| $coeffs$ | - An array containing starting values for the coefficients that are included in the model in the order full AR, subset AR, full MA, subset MA, and the mean of X . |
| it | - An integer containing the number of iterations to allow in the estimation procedure. If it is negative, then $-it$ iterations are allowed and the values of the coefficients for the successive iterations are displayed on the screen. If it is 1, then SEASEST only evaluates $rvar$ and sds . |
| eps | - A real scalar containing a convergence criterion. If the maximum value of successive iterates differs by less than eps , SEASEST judges that the algorithm has converged. |

- nb** - An integer containing the number of back forecasts to use in determining initial values in the recursion used in evaluating the sum of squares of residuals function (≥ 0).

Output:

- coeffs** - An array containing the final values reached for the parameters in the iterative process. **coeffs** is not changed from input if **it**=1.
- rvar** - Estimate of error variance.
- ier** - An integer variable indicating whether or not convergence was achieved (0 means yes, 1 means no), if a singular matrix was encountered (2), or whether the algorithm could not continue even though convergence was not reached (3 or 4). If this final alternative happens, different starting values or convergence criteria may lead to convergence.
- sds** - An array containing the standard errors of the estimates.
- e** - An optional argument which if present is an array of length **n** containing the one step ahead prediction errors corresponding to the **n** values of **X**.

Remarks:

1. In the notation of Chapter 3, we have

$$\text{ords} = \langle p, P, q, Q, M \rangle$$

2. $\text{Workspace} = 3(n + \text{nb}) + 5K + 3\text{maxp} + \text{maxq} + K(n + \text{nb}) + 2(K + 1)^2$, where K is the sum of the elements of **ords**, and **maxp** and **maxq** are the largest AR and MA lags in the model when it is expressed as an ARMA model.

SEASPREP Command

Purpose:

Predict values of a multiplicative subset ARIMA process given data X of length n .

Format:

```
SEASPREP(X,n,ords,coeffs,lags,rvar,tf,tl,hl,conf,
xp,xpl,xpu,ier)
```

Input:

- x,n** - The data set and its length.
- ords** - An array of length 8 containing the full and subset AR orders, followed by the full and subset MA orders, followed by a 1 if a constant term is in the model or a 0 if it is not, followed by the number of first differences in the model, the number of *S*th differences in the model, and finally the value of *S*.
- lags** - An array containing the lags (if any) in the model. If both the subset AR and MA orders are zero, no array called **lags** need be formed, but an argument must be included.
- coeffs** - Values for full AR, subset AR, full MA, and subset MA coefficients, followed by the constant if there is one and the values of *m* and λ for the power transform.
- rvar** - Real scalar containing the value of noise variance.
- tf,tl** - Integers containing the prediction origins to use. The values must be at least $\text{maxp} + \text{maxq} + 1$ (*maxp* and *maxq* are the largest AR and MA lags in the expanded version of the model) and at most *n*, and **tf** must be less than or equal to **tl**.
- hl** - Integer containing the maximum number of steps ahead to forecast from each origin.
- conf** - Real scalar containing the confidence level for the probability limits to be placed on the forecasts ($0 < \text{conf} < 1$).

Output:

- xp** - Array of length $(\text{tl} - \text{tf} + 1)\text{hl}$ containing the forecasts. The first *hl* elements are from origin **tf**, the next *hl* are from origin **tf**+1, etc.
- xpl** - Array containing the lower probability limits on the corresponding elements of **xp**.
- xpu** - Array containing the upper probability limits on the corresponding elements of **xp**.
- ier** - Integer variable indicating whether SEASPREDD finished without error (0) or an illegal power transform was requested (1).

Remarks:

1. In the notation of Chapter 3, we have

$$\text{ords} = \langle p, P, q, Q, M, d, D, S \rangle$$

2. $\text{Workspace} = 5(\text{maxp} + \text{maxq} + 1) + 3n + h1 + 3(t1 - tf + 1)h1$, where the values of maxp and maxq are largest AR and MA lags when the model is expressed as an ARMA model.

SIN Command

Purpose:

Calculate sines of various kinds.

Format:

1. $y = \text{SIN}(x)$
2. $Y = \text{SIN}(X, n)$
3. $Y = \text{SIN}(n, a, p)$

Remarks:

1. In the first form, the output real scalar variable y is the sine of the real scalar input (expressed in radians) x .
2. In the second form, the output array Y of length n contains the sines of the corresponding values in the input array X , which must be expressed in radians.
3. In the third form, the elements of the output array Y of length n (n is an input integer) are given by $Y(i) = a \sin(2\pi(i-1)/p)$, where the real scalars a and p are input and $p \neq 0$.

SINGLEOFF Command

Purpose:

Cancel the effect of SINGLEON.

Format:

SINGLEOFF

Remark:

1. See SINGLEON.

SINGLEON Command

Purpose:

Signal TIMESLAB to henceforth wait for a carriage return before invoking any command in a macro; that is, execute macros in single step mode.

Format:

SINGLEON

SORT Command

Purpose:

Form an array consisting of the elements of another array in ascending order.

Format:

Y= SORT(X,n)

Input:

- | | |
|---|---|
| X | - Array of length n containing the data to be sorted. |
| n | - Integer containing the length of X. |

Output:

- | | |
|---|---|
| Y | - Array of length n containing the sorted data. |
|---|---|

SPEAKER Command

Purpose:

Make the PC's speaker sound for a specified length of time in a specified frequency.

Format:

1. `SPEAKER(freq,time)`
2. `SPEAKER(freqs,times,n)`

Remarks:

1. The first form turns on the speaker in frequency `freq` Hertz for time hundredths of a second. Both arguments are integers.
2. The second form takes the two input arrays `freqs` and `times` of length `n` and successively turns on the speaker for `times(i)` hundredths of seconds in frequency `freqs(i)` Hertz.
3. The middle C major scale is 523, 587, 659, 698, 784, 880, 988, 1047 Hertz.
4. Frequencies must be between 21 and 65,535, and times must be between 1 and 65,535.

SPEAKEROFF Command

Purpose:

Cancel the effect of `SPEAKERON`.

Format:

`SPEAKEROFF`

SPEAKERON Command

Purpose:

Henceforth, any time a line is executed during a macro, the speaker beeps.

Format:

SPEAKERON

SUBMNS Command

Purpose:

Calculate and/or subtract and/or add seasonal means for a data set.

Format:

$Y = \text{SUBMNS}(X, n, d, \text{xbar}[, \text{iopt}])$

Input:

- X - Array of length n containing data to be operated upon.
- n - Integer containing the length of X (>0).
- d - Integer containing the period of the seasons (>0).
- iopt - Optional argument which if present is an integer indicating what operation is to be performed. The following options are available:
 - 0 Calculate seasonal means but don't subtract them.
 - 1 Calculate and subtract seasonal means.
 - 2 Add inputted seasonal means xbar to the data (this allows for the inverse operation of $\text{iopt}=1$).
- xbar - (Only if $\text{iopt}=2$) Array of length d containing seasonal means to add to data.

Output:

- Y - (Unless $\text{iopt}=0$) Array of length n containing the result of subtracting or adding seasonal means.
- xbar - (Unless $\text{iopt}=2$) Array of length d (if $d>1$) or a real scalar variable (if $d=1$) containing seasonal means.

Remarks:

1. The argument iopt not being present is equivalent to $\text{iopt}=1$.
2. $\text{Workspace}=n+d$.

SWEEP Command

Purpose:

To sweep a matrix.

Format:

1. $B = \text{SWEEP}(A, n, k1, k2, ier)$
2. $B = \text{SWEEP}(A, n, ind, m, ier)$

Input:

- A - Array of length $n*n$ containing the matrix to be swept.
- n - Integer containing the size of the matrix to be swept.
- $k1, k2$ - In the first form, A is swept on diagonals $k1$ through $k2$.
- ind, m - In the second form, A is swept on the m diagonals whose indices are in the array ind .

Output:

- B - Array of length $n*n$ containing the result of the sweeping.
- ier - Integer variable containing an indicator of whether or not a zero diagonal was encountered during the sweeping (0 means no, 1 means yes).

Remarks:

1. In either form, the next diagonal to be swept on is the one having the largest absolute value of those remaining.
2. If $ier=1$, then B is not formed.
3. $\text{Workspace} = n^2 + K$, where K is the number of diagonals on which A is swept.

TEXTCOLOR Command

Purpose:

Allow the user to set the foreground and background colors in the text mode.

Format:

TEXTCOLOR(back,fore)

Input:

- back
- Integer containing the number of the color to be used for the background. The choices are those numbered 0 through 7 in the table of colors below.
- fore
- Integer containing the number of the color to be used for the foreground. The choices are those numbered 0 through 15 in the table of colors below.

Available Colors in the Text Mode

Number	Color	Number	Color	Number	Color
1	Blue	6	Yellow-brown	11	Light blue-green
2	Green	7	White	12	Light red
3	Blue-green	8	Gray	13	Light purple
4	Red	9	Light blue	14	Yellow
5	Purple	10	Light green	15	Bright white

Remarks:

1. To get colors other than white on black, the user must have booted the PC with a file called CONFIG.SYS on the booting directory that contains the line DEVICE=ANSI.SYS and ANSI.SYS (which comes with DOS) must also be on the booting directory.
2. TIMESLAB starts with white on blue.

TIME Command

Purpose:

To obtain the number of seconds since midnight. This allows the user to time TIMESLAB operations.

Format:

TIME(t)

Remark:

1. τ is an output real scalar variable containing the time since midnight in seconds. **TIME** can only return the time to the nearest .05 second (approximately).

TOEPL Command

Purpose:

To form a symmetric Toeplitz matrix.

Format:

$A = \text{TOEPL}(R, R_0, n)$

Input:

- R - (If $n > 1$) Array of length $n-1$ containing the second through n th elements of the first row of A .
- R_0 - Real scalar containing the value for the diagonal of A .
- n - Integer ($1 \leq n \leq 100$) containing the size of the resulting matrix.

Output:

- A - Array of length $n*n$ containing the desired matrix.

Remarks:

1. Even though the first argument is ignored if $n=1$, something must be included in the argument list.
2. $\text{Workspace} = n^2$.

TRANS Command

Purpose:

Find the transpose of a matrix.

Format:

$Y = \text{TRANS}(X, n, m)$

Input:

- X - Array of length $n*m$ containing the matrix to be transposed.
- n,m - Integers containing the dimensions of X.

Output:

- Y - Array of length $n*m$ containing the transposed matrix.

Remarks:

1. The first n elements of X will be considered to be its first column, the next n will be the second column, and so on.
2. Workspace= nm .

TYPE2 Command

TIMESLAB has a wide variety of assignment statements and arithmetic operations on integers, reals, and arrays. See Section B.4.3 for details on how these operations are performed.

TYPE4 Command

Purpose:

To define an array by listing its elements or concatenating reals and/or arrays. Some examples are (see Section B.4.2 for more information):

1. alpha=<1,2,3,4,5>
2. X=<1,2,Y,Z,X,6.53>

WHILE Command

Purpose:

To provide a WHILE...END construct in macros.

Format:

WHILE(var1.exp.var2)

Remark:

1. The notation `var1.exp.var2` represents an immediate or variable integer or real scalar followed by a decimal point, followed by one of the expressions `lt`, `le`, `eq`, `ge`, or `gt` followed by another decimal point and another immediate or variable integer or scalar. All of this must occupy no more than 15 characters. If the expression is false, the next line of the macro to be executed is after the corresponding `END`. Otherwise it is the next line after the `WHILE`.

WINDOW Command

Purpose:

Calculate nonparametric spectral density estimates for univariate data.

Format:

`f=WINDOW(rho,R0,M,Q,iopw[,n,c])`

Input:

- | | |
|-------------------|---|
| <code>rho</code> | - Array of length <code>M</code> (if <code>iopw</code> is between 1 and 5) or length <code>n-1</code> if <code>iopw</code> is between 6 and 8 containing autocorrelations. |
| <code>R0</code> | - Real scalar containing the sample variance (> 0). |
| <code>M</code> | - Integer (> 0) containing scale parameter. |
| <code>Q</code> | - Number of frequencies between 0 and 1 at which to calculate spectra (the greatest prime factor of <code>Q</code> must be ≤ 23 , and the product of its square-free prime factors must be ≤ 210). |
| <code>iopw</code> | - Integer containing the number of the window to be used in the estimation procedure (see the table below for the list of choices). |
| <code>n</code> | - (If either <code>iopw</code> is between 6 and 8 or the factor for determining confidence intervals is desired) Integer containing the length of the data set being analyzed. |

Windows in the WINDOW Command

ioptw	Window	ioptw	Window
1	Truncated periodogram	5	Bohman
2	Bartlett	6	Daniell
3	Tukey	7	Bartlett-Priestley
4	Parzen	8	Parzen-Cogburn-Davis

Output:

- f - Array of length $[Q/2]+1$ containing the spectral estimator at the frequencies $(j-1)/Q$, $j=1,...,[Q/2]+1$.
- c - If this argument is present, it is a real scalar variable that can be used to find 95% confidence intervals for the true spectral density. The interval at the i th frequency would be from $f(i)/c$ to $f(i)*c$.

Remark:

1. Workspace= $M+3Q$.

WN Command

Purpose:

Generate realizations from a white noise series.

Format:

`X=WN(seed,n,iopt)`

Input:

- seed - Real scalar containing the seed for the random number generator.
- n - Integer containing the length of the desired realization.
- iopt - Integer containing the number of the distribution to use (see the table below for a list of choices).

Distributions in the WN Command

iopt	Distribution	iopt	Distribution
1	$N(0,1)$	5	Standard Cauchy
2	$U(0,1)$	6	Extreme value
3	Unit exponential	7	Lognormal
4	Logistic	8	Double exponential

Output:

x - Array of length **n** containing the realization.

Remarks:

1. Leaving out **iopt** is the same as using **iopt=1**.
2. Samples from each distribution are generated by transforming uniforms or normals. The uniforms are generated via the multiplicative congruential generator using multiplier 7^5 and modulus $2^{31}-1$ (see Lewis et al. (1969)). The normals are generated using the Marsaglia (1962) polar method.
3. **Workspace=n**.

APPENDIX D

The Data Sets

In this appendix we list the values for all of the data sets that are analyzed in this book. These listings are exactly how the data sets appear on distribution diskette number 2. Next to the heading for each data set is the name of the file containing the data. Note that the gas furnace data, New York City temperatures and births data, and the critical radio frequencies data are bivariate data sets. The univariate series in these three data sets can be read by the commands (taking the critical radio frequencies for example):

```
READ(cradfq,x,n)
READ(cradfq,y,n,1)
```

The first ten data sets are Series I–X in Chapter 1. Note that several of the data sets have been coded by linear transformations. This has no effect on correlations or standardized spectral density plots.

The index of this book contains a list of the page numbers where each data set is discussed in the book.

Daily California Births: CALFEM.DAT

This data set is an example of daily data.

Female Births in California (1959)

365										
35.	32.	30.	31.	44.	29.	45.	43.	38.	27.	
38.	33.	55.	47.	45.	37.	50.	43.	41.	52.	
34.	53.	39.	32.	37.	43.	39.	35.	44.	38.	
24.	23.	31.	44.	38.	50.	38.	51.	31.	31.	
51.	36.	45.	51.	34.	52.	47.	45.	48.	39.	
48.	37.	35.	52.	42.	45.	39.	37.	30.	35.	
28.	45.	34.	36.	50.	44.	39.	32.	39.	45.	
43.	39.	31.	27.	30.	42.	46.	41.	36.	45.	
46.	43.	38.	34.	35.	56.	36.	32.	50.	41.	

39. 41. 47. 34. 36. 33. 35. 38. 38. 34.
 53. 34. 34. 38. 35. 32. 42. 34. 46. 30.
 46. 45. 54. 34. 37. 35. 40. 42. 58. 51.
 32. 35. 38. 33. 39. 47. 38. 52. 30. 34.
 40. 35. 42. 41. 42. 38. 24. 34. 43. 36.
 55. 41. 45. 41. 37. 43. 39. 33. 43. 40.
 38. 45. 46. 34. 35. 48. 51. 36. 33. 46.
 42. 48. 34. 41. 35. 40. 34. 30. 36. 40.
 39. 45. 38. 47. 33. 30. 42. 43. 41. 41.
 59. 43. 45. 38. 37. 45. 42. 57. 46. 51.
 41. 47. 26. 35. 44. 41. 42. 36. 45. 45.
 45. 47. 38. 42. 35. 36. 39. 45. 43. 47.
 36. 41. 50. 39. 41. 46. 64. 45. 34. 38.
 44. 48. 46. 44. 37. 39. 44. 45. 33. 44.
 38. 46. 46. 40. 39. 44. 48. 50. 41. 42.
 51. 41. 44. 38. 68. 40. 42. 51. 44. 45.
 36. 57. 44. 42. 53. 42. 34. 40. 56. 44.
 53. 55. 39. 59. 55. 73. 55. 44. 43. 40.
 47. 51. 56. 49. 54. 56. 47. 44. 43. 42.
 45. 50. 48. 43. 40. 59. 41. 42. 51. 49.
 45. 43. 42. 38. 47. 38. 36. 42. 35. 28.
 44. 36. 45. 46. 48. 49. 43. 42. 59. 45.
 52. 46. 42. 40. 40. 45. 35. 35. 40. 39.
 33. 42. 47. 51. 44. 40. 57. 49. 45. 49.
 51. 46. 44. 52. 45. 32. 46. 41. 34. 33.
 36. 49. 43. 43. 34. 39. 35. 52. 47. 52.
 39. 40. 42. 42. 53. 39. 40. 38. 44. 34.
 37. 52. 48. 55. 50.

Beveridge Wheat Price Index: BEV.DAT

This series has been analyzed extensively in the past (see Anderson (1971), p. 622, for example).

Beveridge Wheat Price Index, 1500-1869

370
 17 19 20 15 13 14 14 14 14 11 16 19
 23 18 17 20 20 18 14 16 21 24 15 16
 20 14 16 25.5 25.8 26 26 29 20 18 16 22
 22 16 19 17 17 19 20 24 28 36 20 14
 18 27 29 36 29 27 30 38 50 24 25 30
 31 37 41 36 32 47 42 37 34 36 43 55
 64 79 59 47 48 49 45 53 55 55 54 56
 52 76 113 68 59 74 78 69 78 73 88 98
 109 106 87 77 77 63 70 70 63 61 66 78
 93 97 77 83 81 82 78 75 80 87 72 65
 74 91 115 99 99 115 101 90 95 108 147 112
 108 99 96 102 105 114 103 98 103 101 110 109
 98 84 90 120 124 136 120 135 100 70 60 72
 70 71 94 95 110 154 116 99 82 76 64 63
 68 64 67 71 72 89 114 102 85 88 97 94
 88 79 74 79 95 70 72 63 60 74 75 91
 126 161 109 108 110 130 166 143 103 89 76 93

82	71	69	75	134	183	113	108	121	139	109	90
88	88	93	106	89	79	91	96	111	112	104	94
98	88	94	81	77	84	92	96	102	95	98	125
162	113	94	85	89	109	110	109	120	116	101	113
109	105	94	102	141	135	118	115	111	127	124	113
122	130	137	148	142	143	176	184	164	146	147	124
119	135	125	116	132	133	144	145	146	138	139	154
181	185	151	139	157	155	191	248	185	168	176	243
289	251	232	207	276	250	216	205	206	208	226	302
261	207	209	280	381	266	197	177	170	152	156	141
142	137	161	189	226	194	217	199	151	144	138	145
156	184	216	204	186	197	183	175	183	230	278	179
161	150	159	180	223	294	300	297	232	179	180	215
258	236	202	174	179	210	268	267	208	224		

Normal White Noise Series: NORMWVN.DAT

This series was formed using the WN command.

Normal White Noise Series

106

.0943	-.2142	.8439	.7062	.5208	.2100
1.4550	.4649	-.4410	-.9391	-.5790	-1.0580
-.6708	.8435	-1.7748	-.8621	-.0782	-.2264
-.5232	-.7443	.4134	-.3300	.2164	-.1358
-.6790	.8799	-1.4526	.9732	.0566	-1.4173
.6477	-.2606	-.4727	-.0500	.8909	.6899
-1.0235	1.2896	-.2868	-.7879	1.7034	.7273
-.5479	-.6860	.7058	.2904	.2679	1.8345
2.5018	-.0287	-1.6743	1.3541	-.9202	1.1854
-1.0682	.0897	-.6558	.0990	-.9942	.5082
.4487	-.1672	2.2177	-.6841	.3455	-.2513
.1213	-.9045	-1.1407	-.2203	-.7520	-.7361
.1348	-.7635	1.5830	.5722	-.7494	-1.5595
1.0255	-.0823	-1.2096	1.7183	-1.0408	-1.0837
.0935	1.8412	.6691	-1.0086	-1.7471	-.6900
1.4420	.2133	-1.3858	-.8704	-1.3716	.9627
-1.1039	-.1556	1.1266	.4647	-.3326	-.0414
2.1232	-.6909	.2319	-.5646		

Annual Rainfall Data: RAINEAST.DAT

This series is typical of many univariate meteorological data sets in that there is very little serial autocorrelation.

Rainfall Eastern US, 1817-1922

106

22.54	-17.46	-26.46	11.54	6.54	-26.46	9.54	-1.46	-33.46	-14.46
6.54	-9.46	6.54	-1.46	-1.46	.54	-6.46	-15.46	-5.46	4.54
-7.46	-2.46	-6.46	-2.46	6.54	1.54	7.54	-9.46	-6.46	9.54
10.54	-2.46	-.46	14.54	-13.46	1.54	-4.46	-5.46	.54	-13.46
5.54	6.54	14.54	-7.46	-3.46	3.54	3.54	-14.46	16.54	-.46

2.54	12.54	6.54	13.54	1.54	-.46	16.54	5.54	7.54	15.54
7.54	15.54	-2.46	4.54	3.54	6.54	9.54	6.54	3.54	2.54
-2.46	7.54	8.54	10.54	.54	.54	3.54	-6.46	-12.46	-5.46
3.54	5.54	-8.46	-7.46	-4.46	.54	3.54	-9.46	3.54	1.54
-.46	-12.46	-5.46	-11.46	4.54	5.54	-3.46	-9.46	6.54	-6.46
-11.46	-9.46	9.54	3.54	-2.46	.54				

Artificial Series: ARTIF.DAT

See Section 1.5.2 for a description of how this data set was formed.

Artificial Time Series

100

12.5971	12.2084	13.0362	10.1389	8.8044	7.5840
9.7160	11.4201	13.0817	14.1539	14.0661	12.5436
10.7436	10.2038	9.7150	9.7878	10.0856	11.2560
12.5911	13.5058	14.8224	14.0889	13.0398	11.1967
10.3866	10.0452	9.1542	12.0071	13.9843	14.1276
15.7731	15.1213	13.4067	10.9186	8.1324	7.5325
8.2933	11.5182	13.5241	14.6620	17.4796	17.6391
15.4382	12.9454	11.2046	10.8988	12.0149	15.5140
19.5596	20.1978	18.5038	19.2840	16.5496	15.9002
13.0752	12.6914	12.6856	14.5315	15.5767	17.9975
19.2265	18.5547	19.3905	16.4114	15.6677	15.4860
15.8087	16.1202	16.8718	18.4903	18.6455	17.8001
16.9917	14.7829	15.3884	15.4088	15.1301	15.2064
16.5537	20.0116	19.5788	21.2045	20.3659	18.6175
17.0249	18.1552	18.8584	18.8034	18.7778	19.9501
22.4343	22.0911	19.1892	16.8242	14.4648	15.6066
15.4240	17.3307	20.0159	21.5673		

International Airline Data: AIR.DAT

This series is one of the most often analyzed data sets. It is the classic example of the Box-Jenkins multiplicative seasonal model.

Airline Passengers, 1949-60

144

(5X,12F5.0)

1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	506	508	461	390	432

Random Walk Series: RW.DAT

This series was formed by the cumulative sums of a normal white noise series.

Random Walk

200

4.7859	6.4711	5.6384	4.5445	5.2512	5.4320	6.8870
7.3520	7.5554	9.4746	8.8956	7.8375	7.1667	8.0101
8.3214	6.9438	6.8656	6.6391	6.1159	5.3716	5.7850
5.4549	5.6714	5.5356	4.6291	5.5859	4.1332	5.1065
5.1631	3.7457	4.3934	4.1329	3.5110	3.6592	3.4896
5.4150	5.9904	6.8264	6.5395	5.7517	7.4551	8.1824
7.6346	6.9486	6.9439	7.9076	8.1756	10.0101	12.5119
12.4833	10.8089	12.1630	11.2429	12.4282	11.3600	11.4497
10.7938	10.8928	9.8986	10.4068	10.8555	10.6883	12.9060
12.2219	12.2337	13.4497	13.5710	12.6664	11.5258	11.3055
10.5535	9.8174	9.9523	9.1888	10.7718	11.3440	10.5945
9.0351	8.0824	6.9112	5.7016	7.4199	7.0175	6.8965
6.9900	8.8312	9.5003	8.4917	6.7446	6.0545	7.4966
7.7099	6.3240	5.4536	4.0820	5.0447	3.9409	3.7853
4.2093	6.1621	5.8295	5.7880	7.9112	7.2202	7.4522
6.8876	6.9819	6.7677	7.6116	8.3178	8.8387	9.0486
8.6076	7.6684	8.1584	7.4693	5.6946	4.8324	6.6357
6.3943	5.7154	6.5953	6.1226	6.0727	6.9636	7.6535
6.6300	7.9196	6.7333	6.4188	7.1246	7.4150	7.7606
7.5092	8.5348	8.4524	7.4116	6.3278	7.4545	7.9191
8.9072	8.9628	9.1451	9.7785	10.0496	9.8638	10.9912
11.7322	11.7365	13.5885	13.2428	15.0027	15.2795	14.6536
13.8078	13.4062	12.4716	13.3632	13.6777	13.4282	15.5994
16.6311	16.2653	16.0370	15.5422	14.7332	14.2666	13.9666
15.0999	14.1622	14.4308	12.3841	13.1813	12.8510	12.5237
11.9447	11.5721	12.6238	12.2099	13.1526	14.0196	14.3557
11.7496	12.4243	12.8397	11.3166	12.2531	14.1968	15.2011
14.5817	13.9441	14.6993	15.1034	15.4357	16.3698	16.1636
16.5665	15.5200	15.6034	16.5936			

Levels of Lake Erie: ERIEL.DAT

This series has both short and long term rises and falls.

Lake Erie Levels 1921-1970

600

14.763	14.649	15.085	16.376	16.926	16.774	16.490	15.769	15.180	14.383
14.478	14.364	13.928	13.283	13.700	15.465	16.243	16.490	16.243	15.787
15.446	14.649	13.776	13.188	13.283	12.657	12.979	13.909	14.535	14.877
14.858	14.288	13.947	13.416	12.903	13.454	13.491	13.567	13.397	14.440
15.161	15.427	15.693	15.142	14.763	14.288	13.074	12.543	12.239	12.011
12.827	13.567	13.548	13.302	13.188	13.112	12.827	12.201	11.917	11.803
11.157	10.892	11.120	12.600	13.283	13.416	13.340	13.529	13.776	14.307
13.852	13.833	13.169	12.941	13.188	14.383	14.763	15.218	15.161	14.858
14.156	13.586	13.150	14.137	14.231	14.364	13.833	14.478	15.009	15.617

16.148	15.977	15.142	14.592	14.364	14.497	14.554	14.991	15.863	17.932
19.184	19.184	18.956	18.254	17.514	16.660	16.338	16.319	17.457	17.173
17.856	18.596	18.558	18.159	17.685	16.812	16.072	15.332	14.478	14.213
13.738	13.169	12.581	13.245	13.852	14.175	14.288	13.985	13.435	12.884
12.429	12.410	13.397	13.909	13.833	14.099	14.687	14.611	14.383	13.909
13.359	12.296	12.106	11.803	12.353	12.220	12.827	14.250	15.085	14.953
14.440	13.890	13.036	12.201	11.404	11.309	10.987	10.361	10.304	11.347
11.784	11.841	11.841	11.651	11.404	10.873	10.209	10.076	10.247	10.133
10.740	11.556	12.201	12.505	12.732	12.201	12.068	11.290	11.139	11.101
10.342	10.000	11.347	12.770	13.321	13.340	13.188	12.676	12.315	12.049
11.594	11.252	12.467	13.491	13.491	14.156	15.256	15.598	16.034	15.598
14.516	13.435	12.638	12.106	12.144	12.979	11.917	15.066	15.199	15.427
15.427	15.408	14.706	14.137	13.302	12.979	13.036	12.903	13.719	14.801
15.541	15.617	15.503	15.218	14.592	13.852	13.416	13.055	12.315	12.239
12.562	13.890	14.687	15.313	15.408	15.028	14.782	14.213	13.435	13.662
14.288	13.491	13.150	13.586	13.871	14.175	14.099	13.719	13.093	12.562
12.030	12.144	11.860	12.410	12.770	14.630	15.218	15.920	15.882	15.750
15.275	14.801	14.554	14.345	15.104	14.554	14.953	15.958	17.590	18.805
18.805	18.330	17.476	16.812	16.224	15.541	14.744	14.478	14.725	16.319
17.324	17.609	17.211	16.546	15.977	15.427	14.972	14.535	14.213	13.719
15.009	16.319	17.078	17.913	18.159	17.704	17.059	17.268	16.546	16.072
15.996	15.294	15.901	16.300	16.546	17.495	17.666	16.964	16.148	15.427
14.839	14.269	14.118	14.156	14.080	16.414	18.216	19.298	18.824	18.368
17.875	16.869	16.129	15.712	15.617	15.161	16.148	17.609	18.406	18.463
18.254	17.609	16.812	15.712	15.066	14.763	14.877	15.370	15.731	15.996
16.224	16.186	16.015	15.446	14.573	14.080	13.226	12.979	14.459	15.655
15.636	17.154	17.381	17.116	16.736	16.167	15.920	15.408	15.066	15.769
15.787	15.863	17.154	18.178	18.653	18.615	18.406	17.837	17.078	16.471
16.224	16.395	17.400	18.672	19.089	19.829	20.000	19.943	19.526	18.975
18.463	17.268	16.414	16.414	16.755	16.736	17.173	17.647	18.216	18.767
18.539	18.273	17.419	16.679	15.996	15.465	15.408	15.123	16.072	17.685
18.235	18.064	17.818	17.438	16.812	17.116	17.211	17.097	17.495	17.097
18.121	18.748	18.767	18.539	18.102	17.818	17.002	16.357	15.560	15.313
14.782	13.719	14.839	15.825	17.324	17.723	17.685	17.514	16.907	15.863
14.934	14.706	14.383	14.345	14.763	15.958	16.698	16.793	17.230	16.509
15.787	14.991	14.080	14.326	14.497	13.510	13.662	14.042	14.269	14.478
14.972	14.972	14.573	13.776	13.017	12.600	12.296	13.131	13.966	15.028
15.844	15.769	15.237	14.801	14.137	13.890	13.416	13.871	14.478	14.725
14.763	15.806	16.565	17.097	17.306	17.211	16.717	15.787	14.801	14.231
13.966	14.004	15.313	16.357	17.742	17.609	17.249	17.078	16.509	15.465
14.706	14.213	13.662	13.928	14.516	15.180	15.351	15.579	15.446	15.199
14.725	14.383	14.231	13.776	13.150	12.713	13.283	14.478	14.858	14.782
14.307	14.023	13.529	12.922	12.410	11.936	11.784	11.992	12.619	13.662
14.231	14.099	13.833	13.416	12.922	11.974	11.461	11.423	11.860	12.163
13.226	13.966	14.535	14.516	14.231	13.966	13.738	13.226	12.998	13.131
13.700	13.814	14.383	15.047	15.693	15.844	15.712	15.313	14.763	13.548
13.586	14.516	14.383	14.497	14.744	15.806	16.527	16.546	16.717	16.433
15.769	15.275	15.123	15.541	15.825	16.376	16.357	16.907	17.021	17.419
17.533	17.268	16.603	15.825	15.446	15.636	15.693	16.755	16.509	17.723
18.691	19.127	19.564	19.203	18.216	17.211	16.660	16.831	15.769	15.731
15.996	17.021	17.552	17.837	17.856	17.571	17.078	16.660	16.433	16.584

Sum of Four Cosine Curves: COS4.DAT

This series was formed by the macro in Example 1.4.

Sum of Four Cosine Curves

144

16.0000	8.5679	-3.5343	6.3660	.3906	-2.1213
14.0981	3.7592	-9.5681	1.0000	-2.7192	-2.1210
17.0981	8.7175	-4.3377	4.6340	-2.2175	-5.5260
10.0000	-.9077	-14.6619	-4.3660	-8.1944	-7.5390
11.9019	3.9011	-8.6281	1.0000	-5.0847	-7.5392
8.9019	-1.0573	-13.8585	-2.6339	-5.5864	-4.1342
16.0000	8.5679	-3.5343	6.3660	.3906	-2.1212
14.0981	3.7592	-9.5681	1.0000	-2.7192	-2.1210
17.0981	8.7175	-4.3377	4.6340	-2.2175	-5.5260
10.0000	-.9077	-14.6619	-4.3660	-8.1944	-7.5390
11.9020	3.9011	-8.6281	1.0000	-5.0847	-7.5392
8.9020	-1.0573	-13.8585	-2.6340	-5.5864	-4.1342
16.0000	8.5678	-3.5342	6.3660	.3906	-2.1212
14.0981	3.7592	-9.5681	1.0000	-2.7192	-2.1210
17.0981	8.7175	-4.3377	4.6340	-2.2175	-5.5260
10.0000	-.9078	-14.6619	-4.3660	-8.1944	-7.5389
11.9020	3.9011	-8.6281	1.0000	-5.0847	-7.5392
8.9020	-1.0573	-13.8585	-2.6339	-5.5864	-4.1342
16.0000	8.5678	-3.5342	6.3660	.3905	-2.1212
14.0981	3.7592	-9.5681	1.0000	-2.7192	-2.1210
17.0981	8.7175	-4.3377	4.6340	-2.2174	-5.5259
10.0001	-.9078	-14.6619	-4.3660	-8.1944	-7.5390
11.9020	3.9009	-8.6281	1.0000	-5.0847	-7.5392
8.9019	-1.0575	-13.8584	-2.6339	-5.5864	-4.1342

Gas Furnace Data: GASFURN.DAT

This bivariate time series has been analyzed by both Box and Jenkins (1970) and Jenkins and Watts (1968).

Input Gas Rate

296

-.11	.00	.18	.34	.37	.44	.46	.35
.13	-.18	-.59	-1.05	-1.42	-1.52	-1.30	-.81
-.47	-.19	.09	.44	.77	.87	.88	.89
.99	1.26	1.77	1.98	1.93	1.87	1.83	1.77
1.61	1.26	.79	.36	.12	.09	.33	.64
.96	1.41	2.67	2.83	2.81	2.48	1.93	1.49
1.21	1.24	1.61	1.90	2.02	1.82	.54	.12
.01	.16	.67	1.02	1.15	1.15	1.11	1.12
1.22	1.26	1.16	.91	.62	.25	-.28	-1.08
-1.55	-1.80	-1.83	-1.46	-.94	-.57	-.43	-.58
-.96	-1.62	-1.88	-1.89	-1.75	-1.47	-1.20	-.93
-.52	.04	.79	.94	.93	1.01	1.14	1.20
1.05	.60	-.08	-.31	-.29	-.15	-.11	-.19
-.25	-.23	-.01	.25	.33	.10	-.42	-1.14

-2.28	-2.59	-2.72	-2.51	-1.79	-1.35	-1.08	-.91
-.88	-.88	-.80	-.54	-.42	-.27	.00	.40
.84	1.28	1.61	1.75	1.68	1.49	.99	.65
.58	.58	.63	.75	.90	.99	.97	.79
.40	-.16	-.55	-.60	-.42	-.19	-.05	.06
.16	.30	.52	.57	.56	.57	.59	.67
.93	1.34	1.46	1.35	.77	.22	-.24	-.71
-1.10	-1.27	-1.17	-.68	.03	.56	.64	.48
.11	-.31	-.70	-1.05	-1.22	-1.18	-.87	-.34
.06	.08	.00	.00	.21	.56	.78	.86
.92	.86	.42	-.34	-.96	-1.81	-2.38	-2.50
-2.47	-2.33	-2.05	-1.74	-1.26	-.57	-.14	-.02
-.05	-.14	-.28	-.53	-.87	-1.24	-1.44	-1.42
-1.17	-.81	-.63	-.58	-.63	-.71	-.85	-1.04
-1.35	-1.63	-1.62	-1.15	-.49	-.16	-.01	-.09
-.62	-1.09	-1.52	-1.86	-2.03	-2.02	-1.96	-1.95
-1.79	-1.30	-1.03	-.92	-.80	-.87	-1.05	-1.12
-.88	-.40	.19	.66	.71	.61	.50	.60
.94	1.22	1.25	.82	.10	.03	.38	.92
1.03	.87	.53	.09	-.46	-.75	-.95	-1.03
-.93	-.64	-.42	-.28	-.16	-.03	.10	.25
.28	.00	-.49	-.76	-.82	-.74	-.53	-.20
.03	.20	.25	.19	.13	.02	-.18	-.26

Output C02

296

53.8	53.6	53.5	53.5	53.4	53.1	52.7	52.4	52.2	52.0	52.0	52.4
53.0	54.0	54.9	56.0	56.8	56.8	56.4	55.7	55.0	54.3	53.2	52.3
51.6	51.2	50.8	50.5	50.0	49.2	48.4	47.9	47.6	47.5	47.5	47.6
48.1	49.0	50.0	51.1	51.8	51.9	51.7	51.2	50.0	48.3	47.0	45.8
45.6	46.0	46.9	47.8	48.2	48.3	47.9	47.2	47.2	48.1	49.4	50.6
51.5	51.6	51.2	50.5	50.1	49.8	49.6	49.4	49.3	49.2	49.3	49.7
50.3	51.3	52.8	54.4	56.0	56.9	57.5	57.3	56.6	56.0	55.4	55.4
56.4	57.2	58.0	58.4	58.4	58.1	57.7	57.0	56.0	54.7	53.2	52.1
51.6	51.0	50.5	50.4	51.0	51.8	52.4	53.0	53.4	53.6	53.7	53.8
53.8	53.8	53.3	53.0	52.9	53.4	54.6	56.4	58.0	59.4	60.2	60.0
59.4	58.4	57.6	56.9	56.4	56.0	55.7	55.3	55.0	54.4	53.7	52.8
51.6	50.6	49.4	48.8	48.5	48.7	49.2	49.8	50.4	50.7	50.9	50.7
50.5	50.4	50.2	50.4	51.2	52.3	53.2	53.9	54.1	54.0	53.6	53.2
53.0	52.8	52.3	51.9	51.6	51.6	51.4	51.2	50.7	50.0	49.4	49.3
49.7	50.6	51.8	53.0	54.0	55.3	55.9	55.9	54.6	53.5	52.4	52.1
52.3	53.0	53.8	54.6	55.4	55.9	55.9	55.2	54.4	53.7	53.6	53.6
53.2	52.5	52.0	51.4	51.0	50.9	52.4	53.5	55.6	58.0	59.5	60.0
60.4	60.5	60.2	59.7	59.0	57.6	56.4	55.2	54.5	54.1	54.1	54.4
55.5	56.2	57.0	57.3	57.4	57.0	56.4	55.9	55.5	55.3	55.2	55.4
56.0	56.5	57.1	57.3	56.8	55.6	55.0	54.1	54.3	55.3	56.4	57.2
57.8	58.3	58.6	58.8	58.8	58.6	58.0	57.4	57.0	56.4	56.3	56.4
56.4	56.0	55.2	54.0	53.0	52.0	51.6	51.6	51.1	50.4	50.0	50.0
52.0	54.0	55.1	54.5	52.8	51.4	50.8	51.2	52.0	52.8	53.8	54.5
54.9	54.9	54.8	54.4	53.7	53.3	52.8	52.6	52.6	53.0	54.3	56.0
57.0	58.0	58.6	58.5	58.3	57.8	57.3	57.0				

Sales Data: SALES.DAT

This series is discussed extensively in Sections 3.6 and 3.7.

Sales of Company X, 1/1965 - 5/1971

77

154	96	73	49	36	59	95	169	210	278	298	245
200	118	90	79	78	91	167	169	289	347	375	203
223	104	107	85	75	99	135	211	335	460	488	326
346	261	224	141	148	145	223	272	445	560	612	467
518	404	300	210	196	186	247	343	464	680	711	610
613	392	273	322	189	257	324	404	677	858	895	664
628	308	324	248	272							

Wolfer Sunspot Data: WOLFER.DAT

This is another classic data set in the literature of time series analysis (see Morris (1977) for example).

Wolfer Sunspot Numbers, 1749-1963

215

80.9	83.4	47.7	47.8	30.7	12.2	9.6	10.2	32.4	47.6
54.0	62.9	85.9	61.2	45.1	36.4	20.9	11.4	37.8	69.8
106.1	100.8	81.6	66.5	34.8	30.6	7.0	19.8	92.5	154.4
125.9	84.8	68.1	38.5	22.8	10.2	24.1	82.9	132.0	130.9
118.1	89.9	66.6	60.0	46.9	41.0	21.3	16.0	6.4	4.1
6.8	14.5	34.0	45.0	43.1	47.5	42.2	28.1	10.1	8.1
2.5	0.0	1.4	5.0	12.2	13.9	35.4	45.8	41.1	30.4
23.9	15.7	6.6	4.0	1.8	8.5	16.6	36.3	49.7	62.5
67.0	71.0	47.8	27.5	8.5	13.2	56.9	121.5	138.3	103.2
85.8	63.2	36.8	24.2	10.7	15.0	40.1	61.5	98.5	124.3
95.9	66.5	64.5	54.2	39.0	20.6	6.7	4.3	22.8	54.8
93.8	95.7	77.2	59.1	44.0	47.0	30.5	16.3	7.3	37.3
73.9	139.1	111.2	101.7	66.3	44.7	17.1	11.3	12.3	3.4
6.0	32.3	54.3	59.7	63.7	63.5	52.2	25.4	13.1	6.8
6.3	7.1	35.6	73.0	84.9	78.0	64.0	41.8	26.2	26.7
12.1	9.5	2.7	5.0	24.4	42.0	63.5	53.8	62.0	48.5
43.9	18.6	5.7	3.6	1.4	9.6	47.4	57.1	103.9	80.6
63.6	37.6	26.1	14.2	5.8	16.7	44.3	63.9	69.0	77.8
65.0	35.7	21.2	11.1	5.7	8.7	36.1	80.4	114.4	109.6
88.8	67.8	47.5	30.6	16.3	9.6	33.2	92.6	151.6	136.3
134.7	83.9	69.4	31.5	13.9	4.4	38.0	141.7	190.2	184.8
159.0	112.3	53.9	37.5	27.9					

Lynx Data: LYNX.DAT

This classic data set is analyzed in several sections of Chapter 3.

LYNX DATA

114

269.	321.	585.	871.	1475.	2821.
3928.	5943.	4950.	2577.	523.	98.
184.	279.	409.	2285.	2685.	3409.
1824.	409.	151.	45.	68.	213.
546.	1033.	2129.	2536.	957.	361.
377.	225.	360.	731.	1638.	2725.
2871.	2119.	684.	299.	236.	245.
552.	1623.	3311.	6721.	4245.	687.
255.	473.	358.	784.	1594.	1676.
2251.	1426.	756.	299.	201.	229.
469.	736.	2042.	2811.	4431.	2511.
389.	73.	39.	49.	59.	188.
377.	1292.	4031.	3495.	587.	105.
153.	387.	758.	1307.	3465.	6991.
6313.	3794.	1836.	345.	382.	808.
1388.	2713.	3800.	3091.	2985.	3790.
674.	81.	80.	108.	229.	399.
1132.	2432.	3574.	2935.	1537.	529.
485.	662.	1000.	1590.	2657.	3396.

Magnitude of a Star: STAR.DAT

This data set consists of the brightness of a variable star on 600 successive midnights (see Kedem (1986)). See Figures 3.8 and 3.13 for more information about this data set.

MAGNITUDE OF A STAR

600 (20F3.0)

25	28	31	32	33	33	32	31	28	25	22	18	14	10	7	4	2	0	0	0
2	4	8	11	15	19	23	26	29	32	33	34	33	32	30	27	24	20	17	13
10	7	5	3	3	3	4	5	7	10	13	16	19	22	24	26	27	28	29	28
27	25	24	21	19	17	15	13	12	11	11	10	10	11	12	12	13	14	15	16
17	18	19	19	19	19	20	20	20	20	20	20	20	20	21	20	20	20	20	19
18	17	16	15	13	12	11	10	9	9	10	10	11	12	14	16	19	21	24	25
27	28	29	29	28	27	25	23	20	17	14	11	8	5	4	2	2	2	4	6
9	12	16	19	23	27	30	32	33	34	33	32	30	27	24	20	16	12	9	5
3	1	0	0	1	3	6	9	13	17	21	24	27	30	32	33	33	32	31	28
25	22	19	15	12	9	7	5	4	4	5	5	7	9	12	14	17	20	22	24
25	26	27	27	26	25	24	22	20	18	17	15	14	13	13	12	12	12	13	13
13	14	14	15	15	16	17	17	17	17	18	18	19	19	20	20	21	21	22	22
22	22	22	21	20	19	17	16	14	12	11	9	8	7	8	8	9	10	12	14
17	20	23	25	27	29	30	30	30	29	27	25	22	19	16	12	9	6	4	2
1	1	2	4	7	10	14	17	21	25	29	31	33	34	34	33	31	29	26	22
19	14	11	7	4	2	1	0	1	2	5	7	11	15	19	22	25	28	30	32
32	32	31	29	26	23	21	17	14	11	9	7	6	5	6	6	7	9	11	13
15	18	20	22	23	24	25	25	25	24	24	22	21	19	18	17	16	15	15	14
14	14	14	14	14	14	14	14	14	14	14	15	15	15	15	16	16	17	18	20
21	22	23	23	24	24	24	23	22	21	19	17	15	13	11	9	7	6	6	6
7	8	10	12	15	18	22	24	27	29	31	31	31	31	29	27	24	21	18	14
10	7	5	2	1	0	1	2	5	8	12	15	19	23	27	30	32	34	34	34
32	30	28	24	20	16	13	9	6	3	2	1	1	2	4	6	9	13	17	20

23 26 28 30 31 31 31 29 27 24 22 19 16 13 11 9 8 7 7 7
8 9 11 12 14 16 18 20 21 22 23 23 23 23 23 22 21 20 19 18
18 17 17 16 16 16 16 15 15 15 14 14 13 13 13 13 13 13 14 14
15 16 18 19 21 22 24 24 25 26 26 25 24 23 21 19 16 14 12 9
7 5 5 4 5 6 8 10 13 16 20 23 26 29 31 32 32 32 31 29
26 23 20 16 12 8 6 3 1 0 0 1 3 6 10 13 17 21 25 28
31 33 34 34 33 31 29 26 22 18 15 11 8 5 3 2 2 2 4 5

Mauna Loa Carbon Dioxide Data: MLCO2.DAT

This data consists of 216 monthly measurements of carbon dioxide above Mauna Loa, Hawaii.

Mauna Loa Carbon Dioxide
216

14.88 15.62 16.33 17.59 17.93 17.71 15.92 15.15 14.02 12.83
13.64 14.71 15.62 16.59 16.94 17.77 18.29 18.24 16.67 14.96
14.12 13.58 15.14 15.77 16.62 17.16 17.90 19.21 20.02 19.74
18.15 16.00 14.23 14.07 15.04 16.19 16.97 17.74 18.63 19.43
20.47 19.71 18.78 16.84 15.16 15.56 16.14 17.13 18.06 18.59
19.74 20.63 21.21 20.83 19.55 17.75 16.27 15.62 16.84 17.70
18.80 19.08 20.15 21.49 22.25 21.50 19.67 17.61 16.25 16.17
17.01 18.36 19.37 19.93 20.40 21.65 22.26 22.19 20.49 18.48
17.13 17.02 17.84 18.78 19.55 20.65 21.15 22.31 22.35 22.19
21.53 19.13 17.99 17.70 19.15 19.27 20.22 21.23 22.13 23.30
23.57 23.29 22.36 19.71 17.89 17.54 19.36 20.51 21.80 22.03
22.50 24.00 24.46 23.46 22.19 20.57 18.91 18.81 20.24 21.59
22.15 22.73 23.50 24.52 25.11 25.06 23.62 21.55 19.89 19.80
20.73 22.25 23.73 24.53 25.62 26.58 27.24 26.53 25.63 23.28
22.21 21.67 22.61 24.07 24.91 25.81 26.85 28.07 27.97 27.77
26.44 24.92 23.49 23.50 24.34 25.39 26.46 26.93 27.56 28.23
29.51 29.04 27.87 26.00 24.06 24.20 25.48 26.62 27.30 28.20
28.50 30.22 30.58 29.48 28.56 26.77 25.39 25.72 26.97 28.09
29.16 30.02 30.95 31.95 32.85 32.58 31.30 29.64 28.12 27.67
28.69 29.07 29.84 31.17 31.94 33.20 33.55 32.76 31.79 29.61
27.88 27.85 28.79 30.13 30.64 31.21 31.88 33.13 33.76 33.74
32.05 30.40 28.89 28.55 29.57 30.83

Buffalo Snowfall Data: BUFFSNOW.DAT

This data set has been analyzed frequently, not so much from a time series analysis point of view, but from a probability density estimation point of view (see Problem C1.24, Parzen (1979), and Silverman (1986), p. 44).

Yearly Snowfall in Buffalo 1910-1972
63

126.4 82.4 78.1 51.1 90.9 76.2 104.5 87.4 110.5 25.0
69.3 53.5 39.8 63.6 46.7 72.9 79.6 83.6 80.7 60.3
79.0 74.4 49.6 54.7 71.8 49.1 103.9 51.6 82.4 83.6
77.8 79.3 89.6 85.5 58.0 120.7 110.5 65.4 39.9 40.1
88.7 71.4 83.0 55.9 89.9 84.8 105.2 113.7 124.7 114.5
115.6 102.4 101.4 89.8 71.5 70.9 98.3 55.5 66.1 78.4

120.5 97.0 110.0

Critical Radio Frequencies: CRADFQ.DAT

The first univariate series in this bivariate data set consists of the median noon hour value of what are called critical radio frequencies each month in Washington, D.C., for the period May 1934 to April 1954. These frequencies reflect the highest radio frequency that can be used for broadcasting. See Siddiqui (1962) for further discussion of these data. The second univariate series consists of the total number of sunspots for the corresponding months. See Sections 3.8 and 3.9 for a discussion of these series.

Critical Radio Frequencies (fOF2)

240

5.40	5.40	5.10	4.80	5.50	6.10	6.40	6.80
6.40	6.00	6.10	6.10	5.30	5.50	5.70	5.00
5.80	9.00	10.40	9.90	10.70	9.70	8.80	8.10
8.40	6.40	6.70	7.40	9.10	11.90	12.50	11.80
11.10	13.10	12.30	9.70	7.70	7.60	8.10	8.90
10.40	13.00	13.20	12.30	11.50	12.80	12.20	10.00
8.10	7.30	7.40	7.90	9.60	12.10	13.10	12.40
11.10	11.70	10.30	9.30	8.00	7.40	6.90	7.60
8.90	12.20	11.40	10.30	9.50	9.60	9.80	8.40
7.40	6.70	6.40	7.10	9.20	10.70	10.60	9.30
9.00	8.90	8.10	7.20	5.70	5.80	5.80	6.20
6.90	9.10	9.00	8.70	8.00	7.80	8.70	7.40
6.10	5.30	4.90	5.10	6.00	7.40	7.50	7.00
6.40	6.60	6.20	5.60	5.10	5.20	5.20	4.80
5.20	6.00	6.60	7.10	6.40	6.00	5.90	5.20
4.90	4.70	4.70	5.00	5.60	7.00	6.60	7.10
6.50	6.60	6.20	6.30	5.70	5.80	5.50	5.70
6.40	9.30	9.30	8.60	7.70	9.40	9.70	9.20
6.20	6.30	6.10	7.20	8.60	11.50	12.20	12.00
12.00	13.00	12.50	11.20	10.00	6.90	7.10	7.70
10.60	13.00	13.50	12.60	12.30	11.30	11.20	10.80
8.20	7.60	7.10	6.80	9.20	11.50	12.40	11.70
12.10	12.50	12.50	10.60	8.30	7.00	7.10	7.80
10.10	12.60	13.10	12.00	11.30	11.00	10.50	8.80
7.20	6.80	6.00	6.10	7.00	9.20	9.00	8.60
8.40	8.70	7.80	6.30	6.10	6.00	5.60	5.80
7.00	8.60	9.20	8.80	8.00	7.50	6.50	5.50
4.80	5.20	5.00	5.30	6.00	7.20	7.60	7.30
6.40	6.20	5.20	5.20	4.80	4.80	4.30	4.70
6.00	6.20	6.50	6.30	5.90	5.70	5.50	5.00

Monthly Sunspot Numbers 5/34-4/54

240

15.9850	15.3350	15.4650	15.4150	15.2000	15.2850
15.4350	15.7700	15.9300	16.0250	16.1550	15.6100
16.3650	17.2850	16.6950	16.5050	17.1050	17.6600
18.2100	18.0750	18.1400	18.7150	18.8550	18.7450
17.7300	18.5000	17.6150	19.3500	18.8000	19.4500
20.7700	21.1700	21.6250	21.4250	19.1950	20.4650

20.8350	21.5150	22.2550	21.8850	20.0350	21.2450
18.7200	19.4400	19.9200	20.9600	19.3250	20.0500
21.3700	19.8750	23.2650	20.7850	19.4800	19.9550
21.1100	19.6350	19.0150	18.8700	18.2300	20.4550
20.9150	20.0500	19.8800	20.2900	20.6300	19.4050
18.4050	17.1050	17.5250	17.9700	19.1650	18.0350
17.7200	19.1950	18.3750	20.2750	18.3250	17.7500
17.9200	18.4150	17.2800	17.2250	17.3200	16.6400
16.4750	17.9900	18.3450	18.0000	18.2950	17.3150
16.9200	16.6850	16.7800	17.6400	17.7100	18.0350
16.2500	15.5700	15.8850	16.0100	15.8600	15.9600
16.5350	16.1250	15.6200	16.4450	16.3700	16.3050
15.7050	15.3800	15.6600	15.9700	15.5000	15.3900
15.5100	15.9400	15.1850	15.0250	15.5500	15.0150
15.1250	15.2500	15.2500	15.8350	15.7150	15.8450
15.5400	16.4200	15.9250	15.6350	16.0750	16.6000
16.5300	16.8100	17.1300	16.2950	16.7450	18.4400
17.3000	16.3700	17.3800	19.3100	18.8300	18.7850
19.2450	18.6750	20.8100	20.3600	19.7200	20.1150
21.1900	21.0850	20.7850	21.6700	21.4900	22.4900
25.0650	23.1950	22.8950	24.4400	23.4700	23.1800
21.4000	20.8250	20.4250	19.3050	19.7400	24.4850
23.7000	23.3900	22.1100	22.8950	22.1650	21.8150
19.7900	21.9000	20.9550	24.1150	22.8750	22.3500
20.3100	21.0850	21.2900	21.1900	22.2650	21.5800
22.1750	20.8800	20.0800	19.7400	20.4850	20.6700
20.3100	19.1800	19.5500	19.2600	17.5650	18.0700
17.7400	17.7050	17.9950	17.9950	17.7950	19.6450
20.4250	20.0300	18.0750	18.0500	19.1550	17.5800
17.6200	17.2900	17.0350	16.1350	16.1000	16.4550
16.1700	16.8200	16.9650	17.7450	16.4100	16.1900
16.1050	16.7150	16.3250	15.1950	15.5000	16.3900
15.6250	16.0900	15.4300	16.1750	15.9650	15.4100
15.0800	15.1250	15.0100	15.0250	15.5450	15.0900

LH Data: LH.DAT

This bivariate series consists of measurements of luteinizing hormone (LH) at 10-minute intervals for two different cows (labeled cows 2 and 3 in the original study (see Rahe, et al. (1980)) each at day three of their estrous cycle. See Problem C3.23 for more discussion of this data set.

Cow 2 on Day 3 of Cycle
144

14	15	16	14	12	10	11	16	13	12	19	19	14	13	14	20	16
13	13	20	16	14	13	19	18	12	13	20	17	18	13	20	14	13
11	11	18	14	11	9	20	15	14	11	18	13	13	9	19	14	12
10	17	16	12	11	10	19	16	12	11	17	15	13	13	19	17	11
11	17	14	11	9	9	14	11	9	8	12	11	9	9	13	11	8
9	16	13	11	10	13	13	10	9	12	15	11	11	13	13	11	11
9	12	12	10	10	16	13	12	10	9	12	11	10	9	10	14	13
11	10	8	16	12	12	10	9	9	12	12	10	9	10	9	12	15

12	11	9	10	8	8	12	12											
Cow 3 on Day 3 of Cycle																		
144																		
14	11	11	16	13	12	10	9	20	16	13	13	11	9	16	15	13		
14	10	12	18	13	13	11	12	20	20	16	14	13	21	17	14	14		
12	19	18	16	15	14	16	20	17	14	15	12	18	18	15	13	13		
22	16	15	14	12	20	19	16	14	13	12	21	19	18	15	15	25		
21	17	14	18	19	14	14	14	19	19	17	14	13	21	17	16	14		
16	19	18	14	13	18	18	15	16	12	19	16	14	12	18	18	15		
16	12	14	17	15	13	11	20	17	14	13	14	14	14	16	13	16		
17	15	13	11	16	19	16	13	14	18	17	16	15	14	19	19	15		
13	13	11	17	17	15	13	19											

New York City Temperatures and Births: NYCTB.DAT

This bivariate series consists of average temperatures and the number of babies delivered in New York City for the 168 months from 1946 to 1959.

NYC Monthly Temperatures 1946-59

168								
11.506	11.022	14.405	14.442	16.524	17.918	18.959	18.309	
18.160	16.691	14.480	17.862	12.082	10.558	12.138	14.442	
16.152	17.714	19.015	19.164	17.900	16.933	13.364	11.468	
10.000	10.985	12.993	14.480	16.134	17.862	19.238	19.089	
18.067	15.576	14.926	12.398	12.435	12.416	13.067	15.093	
16.766	18.680	19.796	19.424	17.398	16.877	13.866	12.658	
13.048	11.301	12.063	14.164	16.041	18.067	19.108	18.736	
17.212	16.394	13.922	11.691	11.970	11.970	12.900	14.888	
16.747	17.955	19.201	18.903	17.714	16.097	13.234	12.435	
11.952	11.970	12.639	15.204	16.283	18.699	19.851	18.959	
18.030	15.316	14.126	12.323	12.230	12.230	13.234	14.647	
16.729	18.494	19.312	19.164	18.011	16.338	14.368	12.751	
11.022	12.658	12.844	15.037	16.190	18.253	19.182	18.513	
17.621	16.506	13.717	11.840	10.855	11.636	12.825	15.019	
16.952	17.937	20.000	19.535	17.770	16.283	13.327	10.818	
11.264	12.026	12.230	14.126	15.818	18.420	18.699	18.903	
17.175	16.041	13.866	12.844	10.613	12.082	12.955	15.000	
16.747	18.792	19.312	18.680	18.067	15.632	14.349	12.732	
11.152	10.372	12.639	14.981	16.097	17.602	19.331	18.978	
17.695	15.539	14.182	10.781	11.059	11.152	12.621	14.981	
17.100	18.160	19.015	19.331	18.346	16.115	13.532	12.398	

NYC Monthly Births 1946-1959

168								
26.663	23.598	26.931	24.740	25.806	24.364	24.477	23.901	
23.175	23.227	21.672	21.870	21.439	21.089	23.709	21.669	
21.752	20.761	23.479	23.824	23.105	23.110	21.759	22.073	
21.937	20.035	23.590	21.672	22.222	22.123	23.950	23.504	
22.238	23.142	21.059	21.573	21.548	20.000	22.424	20.615	
21.761	22.874	24.104	23.748	23.262	22.907	21.519	22.025	
22.604	20.894	24.677	23.673	25.320	23.583	24.671	24.454	
24.122	24.252	22.084	22.991	23.287	23.049	25.076	24.037	
24.430	24.667	26.451	25.618	25.014	25.110	22.964	23.981	
23.798	22.270	24.775	22.646	23.988	24.737	26.276	25.816	

25.210 25.199 23.162 24.707 24.364 22.644 25.565 24.062
25.431 24.635 27.009 26.606 26.268 26.462 25.246 25.180
24.657 23.304 26.982 26.199 27.210 26.122 26.706 26.878
26.152 26.379 24.712 25.688 24.990 24.239 26.721 23.475
24.767 26.219 28.361 28.599 27.914 27.784 25.693 26.881
26.217 24.218 27.914 26.975 28.527 27.139 28.982 28.169
28.056 29.136 26.291 26.987 26.589 24.848 27.543 26.896
28.878 27.390 28.065 28.141 29.048 28.484 26.634 27.735
27.132 24.924 28.963 26.589 27.931 28.009 29.229 28.759
28.405 27.945 25.912 26.619 26.076 25.286 27.660 25.951
26.398 25.565 28.865 30.000 29.261 29.012 26.992 27.897

Waves Data: WAVES.DAT

This data set is discussed in Example 3.22.

Force on a Cylinder

320

-.103 .183 .241 .281 .298 .065 .102 .143 -.029 -.083
-.315 -.329 -.258 -.255 .001 .204 .318 .595 .585 .524
.315 .059 .005 -.191 -.545 -.693 -.774 -.760 -.407 -.285
-.002 .251 .318 .534 .460 .247 .190 -.053 -.012 -.053
-.241 -.191 -.248 -.295 -.171 -.154 .018 .217 .183 .298
.227 .112 -.096 -.359 -.447 -.481 -.578 -.417 -.339 -.225
.197 .311 .423 .402 .143 .038 -.002 -.117 -.083 -.161
-.268 -.127 -.123 .180 .375 .308 .399 .173 -.093 -.154
-.393 -.305 -.208 -.046 .406 .621 .389 .220 -.204 -.299
-.356 -.390 -.315 -.326 -.154 -.120 -.174 .089 .278 .328
.440 .352 .096 .059 -.218 -.299 -.346 -.410 -.285 -.275
-.194 .150 .268 .402 .547 .416 .345 .042 -.069 -.093
-.322 -.373 -.329 -.258 -.069 .055 .204 .251 .109 .089
.166 .045 .072 -.063 -.245 -.191 -.157 -.009 .069 .005
.069 .005 -.103 .015 -.100 -.019 .140 .001 .008 -.100
-.262 -.302 -.403 -.272 -.059 -.049 .072 -.016 -.171 -.157
-.258 -.285 -.133 .008 .055 .230 .234 .416 .487 .402
.305 .022 -.150 -.315 -.481 -.504 -.760 -.730 -.521 -.386
-.019 .160 .214 .426 .571 .726 .928 .830 .601 .197
-.191 -.164 -.494 -.706 -.642 -.471 -.181 .008 .177 .507
.591 .618 .460 .284 .271 .112 -.120 -.177 -.187 -.326
-.225 -.265 -.140 -.022 -.127 .112 .251 .301 .409 .237
.109 .005 -.167 -.043 -.073 -.133 .048 -.063 .011 .160
.136 .237 .268 .112 .055 -.106 -.181 -.225 -.356 -.161
-.063 -.076 .092 .109 .123 .288 .227 .349 .500 .180
-.039 -.275 -.450 -.467 -.619 -.457 -.292 -.137 .129 .399
.426 .493 .268 .281 .160 -.127 -.204 -.433 -.605 -.488
-.430 -.059 .244 .551 .884 .638 .146 -.029 -.332 -.352
-.346 -.572 -.339 -.245 -.332 -.238 -.059 .217 .244 .086
-.022 -.154 -.309 -.322 -.444 -.275 -.198 -.241 -.016 .079
.099 .284 .065 .055 .062 -.090 -.026 -.049 -.127 .109
.001 -.009 -.056 -.127 .008 .059 .075 .187 .106 .005

APPENDIX E

Index of Commands, Theorems, Macros, Examples, and Figures

E.1. Commands

All of the TIMESLAB commands are described in Appendix C, while most of them are discussed somewhere else in the book. The page numbers in this list refer to the best description of the use of the commands.

ABORTOFF	Henceforth Macros Don't Abort upon an Error	477
ABORTON	Henceforth Macros Do Abort upon an Error	477
ABS	Absolute Value Command	477
ARCORR	Find Correlations of Univariate AR Process	127
ARCORR2	Find Correlations of Bivariate AR Process	324
ARDT	Difference Equation Command or Simulate AR Data	128
ARDT2	Bivariate Version of ARDT	325
ARFILT	Apply AR Filter	129
ARMACORR	Find Correlations of ARMA Process	141
ARMADT	Simulate ARMA Data	483
ARMAPRED	Find ARMA Predictors	134
ARMASEL	Stepwise ARMA Modeling	226
ARMASP	Find Spectral Density of ARMA Process	487
ARPART	Transform AR Coefficients to Partial	126
ARSP	Find Spectral Density of AR Process	489
ARSP2	Find Spectral Density of Bivariate AR Process	326
ARSPCB	Find Confidence Bands for an AR Spectral Density	253
ARSPPEAK	Estimate Peak Frequencies of an AR Spectral Density	259
BARTTEST	Bartlett's Test for White Noise	173
BATCHOFF	Cancel BATCHON	493
BATCHON	Henceforth No Pauses after Plots	494
BINOM	Find Binomial Coefficients or Probabilities	494
CLEAN	Delete Variables	495

CLS	Clear the Screen	495
COEFFCSD	Standard Errors for ARMA Parameter Estimates	199
COLOR	Set Graphics Colors	496
CORR	Find Sample Correlogram and/or Periodogram	8
CORR2	Estimate Auto- and Cross-Correlations from Data	8
CORRAR	Find AR Parameters from Correlations	128
CORRAR2	Find Bivariate AR Parameters from Correlations	328
CORRARMA	Find ARMA Parameters from Correlations	141
CORRMA	Find MA Parameters from Correlations	139
COS	Find Cosine or Generate Points on a Cosine Curve	13
CROSSP	Find Window Cross-Spectral Estimates	334
CUM	Find Cumulative Sums or Averages of an Array	505
CUMSP	Find Cumulative Spectral Density	22
DELAY	Make TIMESLAB Pause for a Specified Time	506
DENSITY	Nonparametric Probability Density Estimation	43
DIFF	Difference an Array	31
DIST	Find pdf, cdf, or Quantiles of Z , t , χ^2 , or F	379
DIVSDS	Find Seasonal Standard Deviations	33
DOS	Issue DOS Commands from TIMESLAB	511
DOT	Find the Inner Product of Two Arrays	512
DOUBLE	Extend a Cross-Spectral Array by Symmetry	320
DTAR	Fit AR Models to Univariate Data	228
DTARMA	Fit ARMA Models from Data	196
DTFORE	Iterated AR Forecasting	244
ECHO	Henceforth Text on the Screen Echoed to Printer	518
EDIT	Invoke the TIMESLAB Text Editor	518
EIG	Find Eigenvalues and Eigenvectors	519
END	Terminator of WHILE...END Construct in Macros	520
ENDIF	Terminator of IF...ENDIF Construct in Macros	520
ERASE	Erase Part of a Graphics Screen	520
EXP	Exponentiation Command	521
EXTEND	Extend an Array by Inverse Differencing	36
EXTRACT	Extract Part of an Array	523
FFT	Fast Fourier Transform Command	28
FILT	General Linear Filter Command	34
GOTO	Unconditional Branch in Macros	525
GRMENU	Place the Graphics Menu onto a Graphics Screen	526
GS	Gram-Schmidt Decomposition of a Matrix	360
HELP	On-line Help System	527
HIST	Plot Histogram of an Array	527
IF	Conditional Branch in Macros	528
INFO	Display a List of All Defined Variables	529
INFQNT	Plot Informative Quantile Function of an Array	530
INVPOLY	Find Coefficients of the Reciprocal of a Polynomial	531
LABEL	Define a New Label or Plot a String of Characters	531

LENGTH	List the Number of Elements of an Array	533
LINE	Form Values That Fall on a Line	533
LIST	Display the Values of an Array or Set of Scalars	534
LISTM	List a Two- or Three-Dimensional Array	535
LISTSP	List the Values of a Spectral Array	535
LOGE	Natural Logarithm Command	536
MACORR	Find Correlations of an MA Process	536
MACRO	Invoke a Macro or Reinvoke an Interrupted Macro	537
MADT	Simulate Data from Gaussian MA Process	538
MASP	Find Spectral Density of MA Process	538
MAXMIN	Find an Array's Maximum and Minimum	539
MCHOL	Modified Cholesky Decomposition of a Matrix	359
MDEL	Delete Some Rows of a Matrix	540
MINV	Invert a Matrix	541
MMULT	Multiply Matrices	541
MULTPOLY	Find Coefficients of the Product of Polynomials	542
NOTES	User-Definable Help System	543
OVEROFF	TIMESLAB Will Ask Permission to Overwrite	543
OVERON	TIMESLAB Automatically Overwrites	543
PAGE	Reset Printer to Top of Next Page	544
PARCORR	Calculate Partial Correlations and Residual Variances of Data	10
PARTAR	Find Partial Correlations from AR Coefficients	126
PAUSE	Make a Macro Pause and Wait for a Response	545
PLOT	Y Versus X plot	546
PLOT2	Superimpose Two Y Versus X Plots	547
PLOTCSF	Plot Spectral Array on Nonlog Scale	21
PLOTK	Superimpose K Y Versus X Plots	548
PLOTOFF	Switch from Graphics to Text Mode	549
PLOTON	Switch from Text to Graphics Mode	550
PLOTSIZE	Set Graphics Characteristics	550
PLOTSP	Plot Spectral Array on Log Scale	21
POLAR	Find Modulus and Arctangent of Two Arrays	307
POLY	Evaluate the Values of a Polynomial	553
POLYROOTS	Find Roots of a Polynomial	554
PRINT	Send a List of Variables to Printer	554
PRINTER	Set Printer Characteristics	555
PRINTSEL	Specify Printer Type	556
PROMPTOFF	Henceforth Each Command in Macro Not Displayed	556
PROMPTON	Henceforth Each Command in a Macro Displayed	557
PSOFF	Stop Automatic Printing of Graphs	557
PSON	Henceforth All Plots Also Printed	557
QTEST	Q Test for White Noise	174
QUIT	Exit from TIMESLAB and Return to DOS	558
READ	Read an Array from a File	558
RECORD	Keep a Record of a Session onto a File	559

REG	General Purpose Regression Command	394
REPLACE	Replace Specified Elements of an Array	560
RESCREEN	Redisplay a Graphics Screen That Has Been Saved	561
RESTART	Reset Parameters and Restart TIMESLAB	562
REVERSE	Reverse the Elements of an Array	562
ROOTSPOLY	Find Coefficients of a Polynomial from Roots	563
SAVE	Save an Array to a File	564
SAVEESC	Save a Graphics Screen to a File	564
SEASEST	Multiplicative Subset ARMA Estimation	201
SEASPRED	Multiplicative Subset ARMA Prediction	238
SIN	Find Sine or Generate Points on a Sine Curve	13
SINGLEOFF	Stop Single-Step Execution of a Macro	568
SINGLEON	Start Single-Step Execution of a Macro	569
SORT	Sort an Array	569
SPEAKER	Make the Speaker Sound	569
SPEAKEROFF	Stop Speaker Beeping at Each Command in a Macro	570
SPEAKERON	Make Speaker Beep at Each Command in a Macro	570
SUBMNS	Calculate Seasonal Means	33
SWEEP	Matrix Sweep Command	363
TEXTCOLOR	Set Colors in Text Mode	572
TIME	Find Number of Seconds since Midnight	573
TOEPL	Form Symmetric Toeplitz Matrix	72
TRANS	Find Transpose of a Matrix	574
TYPE2	Assignment and Arithmetic Operations	575
TYPE4	Define the Elements of or Concatenate Arrays	575
WHILE	Start a WHILE...END Construct	575
WINDOW	Univariate Nonparametric Spectral Estimation	187
WN	Generate White Noise Data	4

E.2. Theorems

Theorem 1.4.1.	Orthogonality of Sinusoids	13
Theorem 1.4.2.	Sinusoidal Decomposition of Data	16
Theorem 1.4.3.	Standardizing the Periodogram	19
Theorem 1.4.4.	Relations among $\hat{\rho}$, \hat{R} , \hat{f}	24
Theorem 1.6.1.	Solution of Difference Equations	40
Theorem 2.2.1.	Two Spectral Representations	72
Theorem 2.3.1.	Univariate Filter Theorem	79
Theorem 2.4.1.	Univariate Prediction	87
Theorem 2.5.1.	Properties of Random Walks	90
Theorem 2.5.2.	Invertibility of MA Processes	94
Theorem 2.5.3.	Properties of Autoregressive Processes	101
Theorem 2.5.4.	Exponential Decay of AR Correlations	104
Theorem 2.5.5.	Autoregressive Approximation	105
Theorem 2.5.6.	R and f for an ARMA Process	109

Theorem 2.5.7.	Inverse Parameters for ARMA Models	116
Theorem 2.5.8.	Properties of the General Linear Process	117
Theorem 2.6.1.	Prediction Via the MCD	118
Theorem 2.6.2.	Gaussian Likelihood Function	119
Theorem 2.6.3.	The Levinson Recursion	121
Theorem 2.6.4.	Partial Autocorrelations	123
Theorem 2.6.5.	The MCD of Γ_p for an AR(p)	124
Theorem 2.6.6.	The Schur Test for Stationarity	125
Theorem 2.6.7.	ARMA Prediction Via the MCD	129
Theorem 2.6.8.	ARMA State and Observation Equations	131
Theorem 3.1.1.	Sampling Properties of \bar{X}	161
Theorem 3.1.2.	Positive Definiteness of \hat{R}	165
Theorem 3.1.3.	Properties of \hat{R} and $\hat{\rho}$	166
Theorem 3.1.4.	Properties of the Periodogram	169
Theorem 3.3.1.	Properties of the Window Estimators	190
Theorem 3.4.1.	Properties of MLE's for Time Series	197
Theorem 3.4.2.	Asymptotic Distribution of ARMA MLE's	197
Theorem 3.4.3.	Properties of YWE's	206
Theorem 3.4.4.	Cybenko's Theorem	207
Theorem 3.4.5.	A "Given Data" Levinson Algorithm	208
Theorem 3.4.6.	Properties of Partial Autocorrelations	211
Theorem 3.5.1.	Inconsistency of AIC for an AR(p)	222
Theorem 3.5.2.	Consistent AR Order Determination	222
Theorem 3.8.1.	Conditions for AR(∞) Representation	247
Theorem 3.8.2.	Properties of CAT	248
Theorem 3.8.3.	Properties of AR Spectral Estimation	249
Theorem 3.8.4.	Confidence Bands for AR Spectra	252
Theorem 3.9.1.	Fisher's Exact Test	256
Theorem 3.9.2.	Estimating a Peak Frequency	258
Theorem 4.1.1.	Spectral Representation of R	305
Theorem 4.1.2.	Range of Coherence Spectrum	308
Theorem 4.1.3.	Two Properties of Coherence	313
Theorem 4.1.4.	Bivariate Filter Theorem	314
Theorem 4.1.5.	Optimal Filters	318
Theorem 4.1.6.	Approximating Filter Coefficients	320
Theorem 4.1.7.	Stationarity of AR	323
Theorem 4.1.8.	Bivariate Levinson Algorithm	327
Theorem 4.2.1.	Properties of \bar{X} and \bar{R}	330
Theorem 4.2.2.	Testing for Independence	333
Theorem 4.2.3.	Properties of Bivariate Periodogram	336
Theorem 4.2.4.	Bivariate Window Estimators	337
Theorem 4.2.5.	Properties of Spectral Quantities	337
Theorem A.1.1.	Modified Cholesky Decomposition	359
Theorem A.1.2.	Gram-Schmidt Decomposition	360
Theorem A.1.3.	Properties of the Sweep Operator	361

Theorem A.1.4.	The Matrix Inversion Lemma	363
Theorem A.2.1.	Fourier Series Approximation	364
Theorem A.2.2.	Properties of Fourier Series	365
Theorem A.2.3.	Spectral Representation of R	368
Theorem A.2.4.	Lebesgue-Stieltjes Integral	368
Theorem A.3.1.	The FFT Recursion	371
Theorem A.4.1.	Mean and Variance of a Linear Function	376
Theorem A.4.2.	Properties of the Multivariate Normal	377
Theorem A.4.3.	Simultaneous Confidence Intervals	379
Theorem A.4.4.	General Prediction Theory	380
Theorem A.4.5.	The Kalman Filter	382
Theorem A.4.6.	Continuous Function Theorem	387
Theorem A.5.1.	Properties of Least Squares Estimators	390
Theorem A.5.2.	Inferences for Normal Errors	391
Theorem A.5.3.	Central Limit Theorem for Regression	394
Theorem A.5.4.	Generalized Least Squares	396
Theorem A.5.5.	Recursive Regression Formulas	403

E.3. Macros

AIC	AIC for AR Processes	221
AIREST	Parameter Estimates for Airline Data	203
ARGAMI	Inverse of Covariance Matrix for AR Process	154
ARMAAIC	AIC for ARMA Process	278
ARSP	Autoregressive Spectral Estimation	288
ARSPSM	Illustrate Properties of AR Spectral Estimation	289
ARTREG	Regression Residuals for Artificial Data	56
AVEPER	Average Periodogram over a Frequency Band	270
BAND	Bandpass Filter Example	147
BAR	Bivariate Autoregressive Modeling	352
BARTCDF	Find CDF of Bartlett Statistic	267
BAUER	Illustrate Bauer's Algorithm	152
BINARY	Form a Binary Series by Clipping	54
BURGYW	Compare Burg and Yule-Walker Estimates	274
CATWN	Significance Level of CAT As White Noise Test	277
CHIPLOT	Plots of χ^2 pdf's	465
CIRCLE	Use PLOT2 to Draw a Circle	461
CIS	Confidence Intervals for the Mean of an AR(1)	261
COHER	Illustrate Coherence Spectrum	348
COHTEST	Test for Zero Coherence	351
CORR2PLT	List and Plot Auto- and Cross-Correlations	346
COS	Illustrate Adding Cosines	49
COSDE	Generate a Cosine Curve Via Difference Equation	61
CROSSP	Calculate Bivariate Cross-Spectra	308
CROSSPE	Estimate Bivariate Cross-Spectra	350

DESCRIBE	Display Univariate Descriptive Statistics	47
DIRICH	Display Dirichlet Kernel	146
DTFORE	Forecasting Via Iterated AR Models	245
EIG	Illustrate Relationship of Eigenvalues and Spectra	143
EXAMPLE	Skeleton for General Macros	448
EXPSM	Choose Exponential Smoothing Parameter	57
FEJER	Display Fejér Kernel	266
FIG41	Construct Figure 4.1	345
FILTEST	Estimate Coefficients of Optimal Filter	344
FILTEX	Illustrate Approximating Filter Coefficients	321
FINDFILT	Approximate Filter Coefficients	320
FISHER	Fisher's Exact Test for Periodicity	286
FISHPV	p -Value for Fisher's Exact Test	286
FPLOT	Plot F pdf's	466
GPAC	Calculate and Display GPAC Array	217
HAT2	Plot Rows of Hat Matrix for Quadratic Regression	462
ID	ARMA Model Identification	279
IDARMA	Illustrate Patterns in ARMA Quantities	150
INCONS	Illustrate Inconsistency of Periodogram	266
INDTEST	Test for Independence of Two Series	349
INNOV	Estimate Innovation Variance	273
LAMBDA	Finding Power Transform for Sales Data	233
LREGAR	Regression with AR Errors for Lynx Data	242
LSAR	Least Squares Estimates for AR Process	213
LYNXCB	AR Spectral Confidence Bands for Lynx Data	287
LYNXPEAK	Estimate Peak Frequency for Lynx Data	288
MEDMN	Example of Simulation Study Macro	457
MISSING	Fill in Missing Data	52
MSERHO	Compare Biased and Unbiased Sample Correlations	262
N01TAB	Form Table of $N(0,1)$ cdf	455
ODE	Illustrate the SPEAKER Command	460
PARZ	Calculate Parzen Lag Weights	148
PERAVE	Illustrate Interpretation of Spectral Density	145
PLAY	"Play" an Array on the Speaker	460
PREWHITE	Prewhiten and Align a Pair of Series	349
PVH	Calculate PVH	275
QLEV	Driver Macro for Effect of m on Q Test	268
QLEVEL	Effect of m in Q Test	269
RANDCOEF	Randomly Generate Coefficients of a Polynomial	149
REG	Ordinary Least Squares Regression	400
REGAR	Regression with AR Errors	241
RW	Generate and Display Random Walks	148
SALESPRD	Predict Sales Data	239
SALESRHO	Find Correlations for Sales Data Models	236
SAWTOOTH	Generate Values on a Sawtooth Function	54

SCATTER	Form Scatterplot for Data Versus Lagged Data	48
SIMFILT	Simulate Data from Transfer Function Model	354
SLOW	Organizing Calculations to Save Time	459
SMOOTH	Example of Moving Average Smoother	56
STDRHO	Find Standard Error of Correlogram	265
TIMECOV	Compare Two Methods of Finding Correlations	55
WINDOW	Display Spectral Windows	271
WINDSP	Window Estimate and Confidence Intervals	271
WINDSP3	Window Estimate for Three Truncation Points	272
WN	The Eight Distributions in the WN Command	46
WNTTEST	Test for White Noise	51

E.4. Examples

Example 1.1.	White Noise Data	46
Example 1.2.	Descriptive Statistics	47
Example 1.3.	Scatterplots	47
Example 1.4.	Sum of Four Cosines	48
Example 1.5.	Testing for White Noise	50
Example 1.6.	Missing Data	52
Example 1.7.	Binary Time Series	53
Example 1.8.	The Sawtooth Function	54
Example 1.9.	Two Methods of Finding Correlations	55
Example 1.10.	Residuals from Regression	56
Example 1.11.	Moving Average Smoother	56
Example 1.12.	Exponential Smoothing	57
Example 2.1.	Eigenvalues of Toeplitz Matrices	143
Example 2.2.	The Spectral Density Function	144
Example 2.3.	The Dirichlet Kernel	146
Example 2.4.	The Bandpass Filter Example	147
Example 2.5.	Random Walks	148
Example 2.6.	Generating Stable Coefficients	149
Example 2.7.	Patterns in ARMA Processes	150
Example 2.8.	Bauer's Algorithm	152
Example 3.1.	Confidence Intervals for the Mean	261
Example 3.2.	MSE for $\hat{\rho}$ and $\tilde{\rho}$	262
Example 3.3.	Standard Error of Autocorrelations	263
Example 3.4.	Inconsistency of the Periodogram	265
Example 3.5.	The Fejér Kernel	266
Example 3.6.	CDF of the Bartlett Statistic	267
Example 3.7.	The Choice of m in the Q Test	268
Example 3.8.	Averaging the Periodogram	270
Example 3.9.	Spectral Windows	270
Example 3.10.	Window Estimators	271
Example 3.11.	Estimating the Innovation Variance	272

Example 3.12.	Yule-Walker and Burg Estimators	273
Example 3.13.	Prediction Variance Horizons	275
Example 3.14.	CAT As a White Noise Test	276
Example 3.15.	AIC for ARMA Processes	278
Example 3.16.	Model Identification	279
Example 3.17.	Fisher's Exact Test	286
Example 3.18.	Confidence Bands for AR Spectra	287
Example 3.19.	Estimating Peak Frequencies	288
Example 3.20.	Autoregressive Spectral Estimation	288
Example 3.21.	Properties of AR Spectral Estimation	289
Example 3.22.	Autoregressive Filtering	290
Example 4.1.	Forming Table 4.1 and Figures 4.1 and 4.2	345
Example 4.2.	Interpreting Coherence	347
Example 4.3.	Testing for Independence	348
Example 4.4.	Nonparametric Spectral Estimation	350
Example 4.5.	Testing for Zero Coherence	351
Example 4.6.	Bivariate Autoregressive Modeling	352
Example B.1.	Forming Tables	453
Example B.2.	A Simulation Study	455
Example B.3.	Arranging Calculations to Save Time	458
Example B.4.	Using the Speaker	460
Example B.5.	Drawing a Circle	461
Example B.6.	Superimposing Plots	462

E.5. Figures

Figure 1.1.	Series I-X	2
Figure 1.2.	Scatterplots	7
Figure 1.3.	Correlograms for Series I-X	9
Figure 1.4.	Partials and Residual Variances for Series I-IX	12
Figure 1.5.	Sums of Cosines and Their Periodograms	18
Figure 1.6.	Periodograms for Series I-IX	20
Figure 1.7.	Cumulative Periodograms for Series I-IX	23
Figure 1.8.	Sample Spectral Density for Series IX	27
Figure 1.9.	Residuals from Regression for Series V	32
Figure 1.10.	Example of Moving Average Smoother	34
Figure 1.11.	Airline Data and Extended Values	37
Figure 1.12.	The Distribution of Differenced Wheat Price Data	45
Figure 1.13.	Example Output from DESCRIBE.MAC	48
Figure 1.14.	Example Output from WNTEST.MAC	51
Figure 1.15.	Two Binary Series	53
Figure 1.16.	The Sawtooth Function	55
Figure 2.1.	Wolfer Sunspot Data	69
Figure 2.2.	Interpreting the Spectral Density Function	77
Figure 2.3.	The Dirichlet Kernel	82

Figure 2.4.	Fourier Series Approximation	85
Figure 2.5.	Five Realizations from a Random Walk Process	91
Figure 2.6.	Examples of Moving Average Processes	99
Figure 2.7.	Examples of Autoregressive Processes	108
Figure 2.8.	Examples of ARMA Processes	113
Figure 2.9.	Airline Data and Transformations	115
Figure 3.1.	Confidence Intervals for the Mean	164
Figure 3.2.	The Féjer Kernel	171
Figure 3.3.	The Inconsistency of the Periodogram	172
Figure 3.4.	The cdf of the Bartlett Statistic	174
Figure 3.5.	Averaging the Periodogram	178
Figure 3.6.	Eight Spectral Windows	185
Figure 3.7.	The Parzen and Tukey Spectral Windows	187
Figure 3.8.	Parzen Window Spectral Estimates for Three Series	195
Figure 3.9.	Prediction Variance Horizon for Airline Data	220
Figure 3.10.	Identifying Power Transformations	232
Figure 3.11.	Sales Data and Box-Jenkins Forecasts	240
Figure 3.12.	Sales Data and Iterated AR Forecasts	245
Figure 3.13.	AR Spectral Estimates for Three Series	249
Figure 3.14.	AR Spectral Confidence Bands for Lynx Data	253
Figure 3.15.	Lynx and Sunspot Data	255
Figure 3.16.	AR Spectral Estimate for Log Lynx Data	260
Figure 3.17.	Correlogram and Confidence Bands for Rainfall Data	265
Figure 3.18.	Yule-Walker and Burg Spectral Estimates	274
Figure 3.19.	The Effect of Autoregressive Filtering	290
Figure 4.1.	Cross-Correlations and Data from a Bivariate Process	303
Figure 4.2.	Auto- and Cross-Spectra for a Bivariate Process	309
Figure 4.3.	Interpreting Coherence Spectra	312
Figure 4.4.	Cross-Correlations and Confidence Bands	334
Figure 4.5.	Coherence and Phase and Confidence Intervals	335
Figure 4.6.	Testing for Zero Coherence	339
Figure 4.7.	Phase and Coherence for Gas Furnace Data	342
Figure B.1.	Using Scaling Information in Plotting Commands	444
Figure B.2.	Displaying the Output of a Simulation Study	457
Figure B.3.	Superimposing Graphs	463

REFERENCES

- Ahlfors, L.V. (1953). "Complex Analysis." McGraw-Hill, New York.
- Akaike, H. (1971). Information theory and an extension of the maximum likelihood principle. Research Memorandum No. 46, Institute of Statistical Mathematics, Tokyo. Published in 2nd Int. Symp. on Inf. Theory, ed. by B.N. Petrov and F. Csaki, Akademiai Kiade, Budapest (1973).
- Anderson, T.W. (1971). "The Statistical Analysis of Time Series." Wiley, New York.
- Anderson, T.W. (1975). Maximum likelihood estimation of parameters of an autoregressive process with moving average residuals and other covariance matrices with linear structure. *Ann. Statist.*, **3**, 1283-1304.
- Ansley, C.F. (1979). An algorithm for the exact likelihood of a mixed autoregressive moving average process. *Biometrika*, **66**, 59-65.
- Ansley, C.F., and Kohn, R. (1985). Estimation, filtering, and smoothing in state space models with incompletely specified initial conditions. *Ann. Statist.*, **13**, 1286-1316.
- Bartlett, M.S. (1950). Periodogram analysis and continuous spectra. *Biometrika*, **37**, 1-16.
- Bartlett, M.S. (1955). "An Introduction to Stochastic Processes with Special Reference to Methods and Applications." Cambridge University Press, London.
- Bauer, F.L. (1955). Ein direktes iterationsverfahren zur Hurwitz zerlegung eines polynoms. *Archiv. Elekt. Übertr.*, **9**, 285-290.

- Berk, K.N. (1974). Consistent autoregressive spectral estimates. *Ann. Statist.*, **2**, 489–502.
- Bhansali, R.J. (1986). Asymptotically efficient selection of the order by the criterion autoregressive transfer function. *Ann. Statist.*, **14**, 315–325.
- Bloomfield, P. (1976). "Fourier Analysis of Time Series: An Introduction." Wiley, New York.
- Bohman, H. (1960). Approximate Fourier analysis of distribution functions. *Ark. Mat.*, **4**, 99–157.
- Box, G.E.P., and Cox, D.R. (1964). An analysis of transformations. *J. Roy. Statist. Soc. Ser. B*, **26**, 211–243.
- Box, G.E.P., and Jenkins, G.M. (1970). "Time Series Analysis, Forecasting, and Control." Holden-Day, San Francisco.
- Box, G.E.P., and Jenkins, G.M. (1973). Some comments on a paper by Chatfield and Prothero and on a review by Kendall. *J. Roy. Statist. Soc. Ser. A*, **135**, 337–345.
- Box, G.E.P., and Pierce, D.A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *J. Amer. Statist. Assoc.*, **65**, 1509–1526.
- Box, G.E.P., and Tiao, G.C. (1975). Intervention analysis with applications to economic and environmental problems. *J. Amer. Statist. Assoc.*, **70**, 70–79.
- Brigham, E.O. (1974). "The Fast Fourier Transform." Prentice-Hall, Englewood Cliffs, N.J.
- Brillinger, D.R. (1975). "Time Series Analysis: Data Analysis and Theory." Holt, Rinehart & Winston, New York.
- Brockwell, P.J., and Davis, R.A. (1985). Applications of a recursive prediction algorithm in time series analysis. Statistics Department, Colorado State University.
- Brockwell, P.J., and Davis, R.A. (1987). "Time Series: Theory and Methods." Springer-Verlag, New York.
- Burg, J.P. (1967). Maximum entropy spectral analysis. 37th Annual International S.E.G. Meeting, Oklahoma City, Oklahoma.
- Chatfield, C., and Prothero, D.L. (1973). Box-Jenkins seasonal forecasting: problems in a case study. *J. Roy. Statist. Soc. Ser. A*, **136**, 295–336.

- Clayton, D.G. (1971). Gram-Schmidt orthogonalization. *Appl. Statist.*, **20**, 335-338.
- Cleveland, W.S. (1972). The inverse autocorrelations of a time series and their applications. *Technometrics*, **14**, 277-293.
- Cogburn, R., and Davis, H.T. (1974). Periodic splines and spectral estimation. *Ann. Statist.*, **2**, 1108-1126.
- Cooley, J.W., and Tukey, J.W. (1965). An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, **19**, 297-301.
- Cybenko, G. (1983). A general orthogonalization technique with applications to time series analysis and signal processing. *Math. Comp.*, **40**, 323-336.
- Davis, H.T. (1980). Comments on Pagano's "Some recent advances in time series," in "Directions in Time Series," ed. by D.R. Brillinger and G.C. Tiao, Institute of Mathematical Statistics, Hayward, California, 301-302.
- Davis, H.T., and Jones, R.H. (1968). Estimation of the innovation variance of a stationary time series. *J. Amer. Statist. Assoc.*, **63**, 141-149.
- DeMeersman, R. (1975). A method for least squares solution of systems with a cyclic rectangular coefficient matrix. *J. Comput. Appl. Math.*, **1**, 51-54.
- Draper, N.R., and Smith, H. (1981). "Applied Regression Analysis." Wiley, New York.
- Durbin, J. (1959). Efficient estimation of parameters in moving average models. *Biometrika*, **46**, 306-316.
- Durbin, J. (1960). The fitting of time series models. *Rev. Inst. Internat. Statist.*, **28**, 233-244.
- Dzhaparidze, K. (1986). "Parameter Estimation and Hypothesis Testing in Spectral Analysis of Stationary Time Series." Springer-Verlag, New York.
- Engel, E.M.R. (1984). A unified approach to the study of sums, products, time-aggregation and other functions of ARMA processes. *J. Time Series Anal.*, **5**, 159-171.
- Ensor, K. (1987). PhD Dissertation, Dept. of Statistics, Texas A&M University.
- Ensor, K., and Newton, H.J. (1987). The effect of order determination on estimating the peak frequency of an autoregressive spectral density. Department of Statistics, Texas A&M University.

- Feller, W. (1948). On the Kolmogorov-Smirnov theorems for empirical distributions. *Ann. Math. Statist.*, **19**, 177-189.
- Findley, D.F. (1978). Limiting autocorrelations and their uses in the identification of nonstationary ARMA models. *Bull. Inst. Math. Statist.*, **7**, 293-294.
- Fisher, R.A. (1929). Tests of significance in harmonic analysis. *Proc. Roy. Soc. Ser. A*, **129**, 54-59.
- Fishman, G.S. (1973). "Concepts and Methods in Discrete Event Digital Simulation." Wiley, New York.
- Friedlander, B., Morf, M., Kailath, T., and Ljung, L. (1979). New inversion formulas for matrices classified in terms of their distance from Toeplitz matrices. *Lin. Alg. Appl.*, **60**, 31-60.
- Fuller, W.A. (1976). "Introduction to Statistical Time Series." Wiley, New York.
- Gentleman, W.M., and Sande, G. (1966). Fast Fourier transforms—for fun and profit. AFIPS, 1966 Fall Joint Computer Conference, *Conf. Proc.*, **28**, 563-578.
- Granger, C.W.J., and Morris, M.J. (1976). Time series modeling and interpretation. *J. Roy. Statist. Soc. Ser. A*, **139**, 246-257.
- Gray, H.L., and Woodward, W. (1986). A new ARMA spectral estimator. *J. Amer. Statist. Assoc.*, **81**, 1100-1108.
- Grenander, U., and Rosenblatt, M. (1957). "Statistical Analysis of Stationary Time Series." Wiley, New York.
- Grenander, U., and Szegö, G. (1958). "Toeplitz Forms and Their Applications." Univ. of California Press, Berkeley.
- Hannan, E.J. (1970). "Multiple Time Series." Wiley, New York.
- Hannan, E.J., and Nicholls, D.F. (1977). The estimation of the prediction error variance. *J. Amer. Statist. Assoc.*, **72**, 834-840.
- Hannan, E.J., and Quinn, B.G. (1979). The determination of the order of an autoregression. *J. Roy. Statist. Soc. Ser. B*, **41**, 190-195.
- Hannan, E.J., and Rissanen, J. (1982). Recursive estimation of mixed autoregressive moving average order. *Biometrika*, **69**, 81-94.
- Harrison, P.J., and Stevens, C.F. (1976). Bayesian forecasting. *J. Roy. Statist. Soc. Ser. B*, **38**, 205-258.

- Hartley, H.O. (1949). Tests of significance in harmonic analysis. *Biometrika*, **36**, 194-201.
- Isserlis, L. (1918). On a formula for the product moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika*, **12**, 134-139.
- Jenkins, G.M., and Watts, D.G. (1968). "Spectral Analysis and Its Applications." Holden-Day, San Francisco.
- Jennrich, R.I. (1977). Stepwise regression, in "Statistical Methods for Digital Computers," ed. by K. Enslein, A. Ralston, and H.S. Wilf, Wiley, New York.
- Jones, R.H. (1980). Maximum likelihood fitting of ARMA models to time series with missing observations. *Technometrics*, **22**, 389-395.
- Jones, R.H., and Brelsford, W.M. (1967). Time series with periodic structure. *Biometrika*, **54**, 403-408.
- Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Trans. ASME, J. Basic Eng.*, **83D**, 35-45.
- Kedem, B. (1980). "Binary Time Series." Marcel Dekker, New York.
- Kedem, B. (1986). Search for periodicities by axis crossings of filtered time series. *Signal Processing*, **10**, 129-144.
- Levinson, N. (1947). The Wiener RMS error criterion in filter design and prediction. *J. Math. Physics*, **25**, 261-278.
- Lewis, P.A.W., Goodman, A.S., and Miller, J.M. (1969). Pseudo-random number generator for the System/360. *IBM Systems J.*, **8**, 136-146.
- Ljung, G.M., and Box, G.E.P. (1978). On a measure of lack of fit in time series models. *Biometrika*, **65**, 297-303.
- Makhoul, J. (1975). Linear prediction: a tutorial review. *Proceedings of the IEEE*, **63**, 561-580.
- Makhoul, J. (1977). Stable and efficient lattice methods for linear prediction. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, **ASSP-25**, 423-428.
- Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E., and Winkler, R. (1984). "The Forecasting Accuracy of Major Time Series Methods." Wiley, New York.

- Marsaglia, G. (1962). Random variables and computers, *Information Theory Statistical Decision Functions Random Processes: Transactions of the Third Prague Conf.*, ed. by J. Kozesnik, Czechoslovak Academy of Sciences, Prague, 499-510.
- Masani, P. (1966). Recent trends in multivariate prediction theory, in *Multivariate Analysis-I*, ed. by P. Khrishnaiah, Academic Press, New York, 351-382.
- Meinhold, R.J., and Singpurwalla, N.D. (1983). Understanding the Kalman filter. *Amer. Statist.*, **37**, 123-127.
- Melard, G. (1984). A fast algorithm for the exact likelihood of autoregressive-moving average models. *Appl. Statist.*, **33**, 104-114.
- Morris, J. (1977). Forecasting the sunspot cycle. *J. Roy. Statist. Soc. Ser. A*, **140**, 437-447.
- Newton, H.J. (1982). Using periodic autoregressions for multiple spectral estimation. *Technometrics*, **24**, 109-116.
- Newton, H.J., and Pagano, M. (1982). The finite memory prediction of a covariance stationary time series. *SIAM J. Sci. Stat. Comp.*, **4**, 330-339.
- Newton, H.J., and Pagano, M. (1983a). A method for determining periods in time series. *J. Amer. Statist. Assoc.*, **78**, 152-157.
- Newton, H.J., and Pagano, M. (1983b). Computing for autoregressions, *Proc. of 15th Interface of Stat. and Comp.*, ed. by J. Gentle, North Holland, Amsterdam, 125-130.
- Newton, H.J., and Pagano, M. (1984). Simultaneous confidence bands for autoregressive spectra. *Biometrika*, **71**, 197-202.
- O'Neill, R. (1971). Function minimization using a simplex procedure. *Appl. Statist.*, **20**, 338-345.
- Pagano, M. (1973). When is an autoregressive scheme stationary? *Communications in Statistics*, **1**, 533-544.
- Pagano, M. (1976). On the linear convergence of a covariance factorization algorithm. *J. Assoc. Comp. Mach.*, **23**, 310-316.
- Pagano, M. (1978). On periodic and multiple autoregressions. *Ann. Statist.*, **6**, 1310-1317.
- Parzen, E. (1957a). On consistent estimates of the spectrum of a stationary time series. *Ann. Math. Statist.*, **28**, 329-348.

- Parzen, E. (1957b). On choosing an estimate of the spectral density function of a stationary time series. *Ann. Math. Statist.*, **28**, 921-932.
- Parzen, E. (1958). On asymptotically efficient consistent estimates of the spectral density function of a stationary time series. *J. Roy. Statist. Soc. Ser. B*, **20**, 303-322.
- Parzen, E. (1961). An approach to time series analysis. *Ann. Math. Statist.*, **32**, 951-989.
- Parzen, E. (1962a). On estimation of a probability density function and mode. *Ann. Math. Statist.*, **32**, 1065-1076.
- Parzen, E. (1962b). "Stochastic Processes," Holden-Day, San Francisco.
- Parzen, E. (1963a). Notes on Fourier analysis and spectral windows. Technical Report No. 48, May 1963, Statistics Department, Stanford University. Reprinted in *Time Series Analysis Papers*, Holden-Day, San Francisco.
- Parzen, E. (1963b). On spectral analysis with missing observations and amplitude modulation. *Sankhā, Ser. A*, **25**, 383-392.
- Parzen, E. (1967). Empirical multiple time series analysis. *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, ed. by L. LeCam and J. Neyman, University of California Press, Vol. I, 305-340.
- Parzen, E. (1971). Efficient estimation of stationary time series mixed schemes. *Bull. Inst. Internat. Statist.*, **44**, 315-319.
- Parzen, E. (1977). Multiple time series: Determining the order of approximating autoregressive schemes. In *Multivariate Analysis - IV*, ed. by P. Khrishnaiah, North Holland, Amsterdam, 283-295.
- Parzen, E. (1979). Nonparametric statistical data modeling. *J. Amer. Statist. Assoc.*, **74**, 105-121.
- Parzen, E. (1981). Time series model identification and prediction variance horizon. *Proc. of Second Tulsa Symposium on Applied Time Series Analysis*, Academic Press, New York, 415-447.
- Parzen, E. (1982). ARARMA models for time series analysis and forecasting. *J. of Forecasting*, **1**, 67-82.
- Parzen, E. (1983a). Informative quantile functions and identification of probability distribution types. *Proc. of 29th Army Conference on the Design of Experiments*, 97-107.

- Parzen, E. (1983b). "Time Series Analysis of Irregularly Observed Data," Proceedings of a Symposium held at Texas A&M University, College Station, Texas, February 10-13, 1983.
- Priestley, M.B. (1981). "Spectral Analysis and Time Series," Vols. 1 and 2. Academic Press, New York.
- Pukkila, T., and Nyquist, H. (1985). On the frequency domain estimation of the innovation variance of a stationary time series. *Biometrika*, **72**, 317-323.
- Quinn, B.G. (1980). The limiting behavior of the autocorrelation function of ARMA processes as several roots of the characteristic equation approach the unit circle. *Communications in Statistics, B*, **9**, 195-198.
- Rahe, C.H., Owens, R.E., Fleeger, J.L., Newton, H.J., and Harms, P.G. (1980). Pattern of plasma luteinizing hormone in the cyclic cow: dependence upon the period of the cycle. *Endocrinology*, **107**, 498-530.
- Rao, C.R. (1973). "Linear Statistical Inference and Its Applications." Wiley, New York.
- Scheffé, H. (1959). "The Analysis of Variance." Wiley, New York.
- Serfling, R.J. (1980). "Approximation Theorems of Mathematical Statistics." Wiley, New York.
- Shibata, R. (1976). Selection of the order of an autoregressive process by Akaike's information criterion. *Biometrika*, **63**, 117-126.
- Shibata, R. (1981). An optimal autoregressive spectral estimate. *Ann. Statist.*, **9**, 300-306.
- Siddiqui, M.M. (1962). Some statistical theory for the analysis of radio propagation data. *J. of Research of Nat. Bur. Stds.*, **66D**, 571-580.
- Silverman, B.W. (1986). "Density Estimation for Statistics and Data Analysis." Chapman and Hall, New York.
- Singleton, R.C. (1969). An algorithm for computing the mixed radix fast Fourier transform. *IEEE Trans. Audio Electron.*, **AU-17**, 93-103.
- Stewart, G.W. (1973). "Introduction to Matrix Computations." Academic Press, New York.
- Stuart, A. (1955). A paradox in statistical estimation. *Biometrika*, **42**, 527-529.

- Tapia, R.A., and Thompson, J.R. (1978). "Nonparametric Probability Density Estimation." Johns Hopkins University Press, Baltimore.
- Tjøstheim, D., and Paulsen, J. (1983). Bias of some commonly-used time series estimates. *Biometrika*, **70**, 389-399.
- Tong, H. (1977). Some comments on the Canadian lynx data. *J. Roy. Statist. Soc. Ser. A*, **140**, 432-436.
- Tsay, R.S. (1986). Behavior of sample means of nearly nonstationary time series. *Proc. of the 1986 Winter Simulation Conference*, 351-355, Society for Computer Simulation, San Diego.
- Tsay, R.S., and Tiao, G.C. (1984). Consistent estimates of autoregressive parameters and extended sample autocorrelation functions for stationary and nonstationary ARMA models. *J. Amer. Statist. Assoc.*, **79**, 84-96.
- Tukey, J.W. (1949). The sampling theory of power spectrum estimates. *Proc. Symp. on Applications of Autocorrelation Analysis to Physical Problems*, NAVEXOS-P-735, 47-67. Office of Naval Research, Department of the Navy, Washington, D.C.
- Ulrych, T.L., and Bishop, T.N. (1975). Maximum entropy spectral analysis and autoregressive decomposition. *Rev. Geo. Space Phys.*, **14**, 183-200.
- Wahba, G. (1980). Automatic smoothing of the log periodogram. *J. Amer. Statist. Assoc.*, **75**, 122-132.
- Welch, P.D. (1967). The use of the fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Trans. on Audio and Electroacoustics*, **AO-15**, 70-73.
- Whittle, P. (1951). "Hypothesis Testing in Time Series Analysis." Ph.D. Thesis, Uppsala. Almqvist & Wiksell, Uppsala.
- Whittle, P. (1963a). "Prediction and Regulation." English Universities Press, London.
- Whittle, P. (1963b). On the fitting of multivariate autoregressions, and the approximate canonical factorization of a spectral density matrix. *Biometrika*, **50**, 129-134.
- Wilson, G. (1969). Factorisation of the covariance generating function of a pure moving average process. *SIAM J. Numer. Anal.*, **6**, 1-7.
- Wilson, G. (1979). Some efficient computational procedures for high order ARMA models. *J. Statist. Comput. Simul.*, **8**, 301-309.

- Woodroffe, M.B., and Van Ness, J.W. (1967). The maximum deviation of sample spectral densities. *Ann. Math. Statist.*, **38**, 1558–1570.
- Woodward, W., and Gray, H.L. (1981). On the relationship between the S array and the Box-Jenkins method of model identification. *J. Amer. Statist. Assoc.*, **76**, 579–587.

AUTHOR INDEX

- Ahlfors, L.V., 102
 Akaike, H., 220
 Andersen, A., 243
 Anderson, T.W., 104, 161, 166, 168,
 189, 213, 295, 363, 394, 580
 Ansley, C.F., 114, 129
 Bartlett, M.S., 167, 173, 298
 Bauer, F.L., 135
 Berk, K.N., 248
 Bhansali, R.J., 223
 Bishop, T.N., 211
 Bloomfield, P., 72
 Bohman, H., 184
 Box, G.E.P., 113, 174, 200, 202, 211,
 230, 232, 233, 243, 343, 344, 585
 Brelsford, W.M., 107
 Brigham, E.O., 26
 Brillinger, D.R., 330, 345
 Brockwell, P.J., 253, 328
 Burg, J.P., 211
 Carbone, R., 243
 Chatfield, C., 232
 Clayton, D.G., 360
 Cleveland, W.S., 115
 Cogburn, R., 183
 Cooley, J.W., 371
 Cox, D.R., 233
 Cybenko, G., 207
 Davis, H.T., 183, 251, 272
 Davis, R.A., 253, 328
 DeMeersman, R., 212
 Draper, N.R., 399
 Durbin, J., 213
 Dzhaparidze, K., 197
 Engel, E.M.R., 156
 Ensor, K., 212, 259
 Feller, W., 173
 Fildes, R., 243
 Findley, D.F., 275
 Fisher, R.A., 256
 Fishman, G.S., 295
 Fleeger, J.L., 591
 Friedlander, B., 120
 Fuller, W.A., 74, 104, 118
 Gentleman, W.M., 371
 Goodman, A.S., 578
 Granger, C.W.J., 156
 Gray, H.L., 216, 218, 254
 Grenander, U., 72, 143, 299
 Hannan, E.J., 213, 222, 223, 224,
 272, 330

- Harms, P.G., 591
Harrison, P.J., 246
Hartley, H.O., 256
Hibon, M., 243

Isserlis, L., 166

Jenkins, G.M., 113, 200, 202, 211,
230, 232, 304, 343, 344, 585
Jennrich, R.I., 397
Jones, R.H., 107, 132, 197, 272

Kailath, T., 120
Kalman, R.E., 382
Kedem, B., 53, 588
Kohn, R., 114

Levinson, N., 89, 120
Lewandowski, R., 243
Lewis, P.A.W., 578
Ljung, G.M., 174
Ljung, L., 120

Makhoul, J., 205, 212
Makridakis, S., 243
Marsaglia, G., 578
Masani, P., 247
Meinhold, R.J., 383
Mélard, G., 196
Miller, J.M., 578
Morf, M., 120
Morris, J., 254, 587
Morris, M.J., 156

Newton, H.J., 107, 118, 129, 205,
243, 252, 259, 591
Nicholls, D.F., 272
Nyquist, H., 272

O'Neill, R., 196
Owens, R.E., 591

Pagano, M., 107, 118, 125, 129,
135, 205, 252, 259

Parzen, E., 42, 45, 52, 167, 183, 189,
194, 213, 218, 223, 243, 254, 259,
299, 300, 340, 380, 589, 591
Paulsen, J., 205
Pierce, D.A., 174
Priestley, M.B., 70, 74, 118, 155, 161,
164, 189, 192, 204, 257, 345, 363
Prothero, D.L., 232
Pukkila, T., 272

Quinn, B.G., 222, 223, 275

Rahe, C.H., 591
Rao, C.R., 362, 376
Rissanen, J., 224
Rosenblatt, M., 299

Sande, G., 371
Scheffé, H., 379
Serfling, R.J., 387
Shibata, R., 222, 251
Siddiqui, M.M., 590
Silverman, B.W., 45, 589
Singleton, R.C., 374
Singpurwalla, N.D., 383
Smith, H., 399
Stevens, C.F., 246
Stewart, G.W., 208
Stuart, A., 212
Szegő, G., 72, 143

Tapia, R.A., 45
Thompson, J.R., 45
Tiao, G.C., 213, 217, 243
Tjøstheim, D., 205
Tong, H., 227
Tsay, R.S., 213, 217, 292
Tukey, J.W., 179, 371

Ulrych, T.L., 211

Van Ness, J.W., 193

Wahba, G., 193
Watts, D.G., 304, 585

- | | |
|-------------------------------|-----------------------------|
| Welch, P.D., 298 | Winkler, R., 243 |
| Whittle, P., 86, 197, 327 | Woodroffe, M.B., 193 |
| Wilson, G., 96, 134, 135, 141 | Woodward, W., 216, 218, 254 |

SUBJECT INDEX

See also Appendix C for an outline and description of the TIMESLAB commands, Appendix D for a listing of the data sets, and Appendix E for an index to the commands, theorems, macros, examples, and figures contained in the book.

- Absolute continuity, 73
- Absolute integrability, 365
- Absolute summability, 73, 117
- Akaike's information criterion (AIC), 219–222, 224, 251, 278, 293, 294, 298
- Aligning time series, 353
- Amplitude, 13
- Amplitude modulated process, 299
- Amplitude spectrum, 307, 310
- Antithetic variables, 295
- Approximate maximum likelihood estimators, 200
- Approximating filter coefficients, 320
- ARARMA modeling, 244
- Arctan2 function, 64
- ARIMA process, 112
- Arithmetic operations in TIMESLAB, 424
- ARMA spectral estimation, 253
- Asymptotically stationary, 158, 299
- Autocorrelation function, 71
- Autocorrelation method, 205
- Autocovariance function
 - Definition, 71
 - Estimation of, 163
 - Properties of, 72, 73
- Autocovariance method, 212
- Autoregressive (AR) process, 98–107
 - Algorithms for, 123–129, 154, 205–213
 - Approximating processes by, 105
 - Bivariate, 322, 327, 339, 352
 - Burg estimators, 211, 273
 - Estimation algorithms, 205–213
 - Estimating peak frequencies, 257–261
 - Examples, 106
 - Filter, 129, 290
 - Least squares estimators, 212
 - Partial autocorrelations of, 106, 126
 - Periodic, 107, 357
 - Prediction of, 105, 130
 - Properties, 101
 - Simulating, 106, 128

- Spectral estimation, 247–253
- Subset AR modeling, 227
- Test for stationarity, 125
- Yule-Walker estimators, 205–210, 273
- Autoregressive-moving average (ARMA) process, 107–112
 - Algorithms for, 129–134, 140–142
 - Approximate MLE's, 200–202
 - Bivariate, 328
 - Exact MLE's, 196–199
 - Examples, 112
 - Identifying, 214–240
 - Likelihood function, 119
 - Markovian representation, 132
 - Partial autocorrelations of, 112
 - Prediction of, 111, 129–131
 - Properties, 109
 - Simulating, 155
- Autoregressive spectral estimation, 247–253
- Backforecasting, 201
- Backshift operator, 84
- Backward prediction coefficients, 123
- Banded matrix, 154
- Bandwidth, 44, 63, 178, 186
- Bandpass filter, 81–83, 145, 147, 155, 157, 180, 181, 290, 366
- Bartlett's formula, 167, 331
- Bartlett's test, 170, 172, 267, 292
- Bayesian forecasting, 346
- Bauer's algorithm, 135, 152
- Biased residual variances, 315
- Binary time series, 53
- Binomial distribution, 61
- Bivariate ARMA process, 328
- Bivariate autoregression, 322, 327, 339, 352
- Bivariate CAT criterion, 340
- Bivariate difference equation, 325
- Bivariate Filter Theorem, 314
- Bivariate general linear process, 329
- Bivariate spectral density estimation, 330
- Bivariate white noise process, 311
- Boxcar function, 205
- Box-Jenkins modeling, 230–240
- Burg algorithm, 211, 273
- CAT criterion, 223, 247, 276
- Central limit theorem, 394
- Cesàro summability, 295
- Characteristic exponent, 189
- Characteristic function, 75, 377
- Characteristic polynomial, 40
- Characteristic value, 189
- Cholesky factors, 63, 96, 118, 358
- Circular shift operator, 305
- Clipped time series, 159
- Coherence spectrum, 307, 310, 313, 355
- Companion matrix, 357
- Complex coherence spectrum, 307
- Conditional expectation, 375, 380
- Confidence bands for spectra, 252, 287
- Confidence intervals
 - For amplitude spectrum, 338
 - For coherence, 338
 - For f , 193, 252, 287
 - For μ , 162, 332
 - For peak frequency, 259
 - For phase spectrum, 338
 - For R and ρ , 167, 332
- Conjugate pairs, 40
- Consistent estimator, 161
- Continuous function theorem, 296, 387
- Convergence of random variables, 386
- Convolutions, 28, 117
- Correlogram, *See* Sample autocorrelation function
- Cospectral density, 307, 310
- Covariance generating function, 79, 134, 140
- Covariance stationary, 70, 301

- Cramer-Wold factorization, 134
 Cross-correlations, 8, 301
 Cross-spectral analysis, 315
 Cross-spectral density, 306
 Cumulants, 166
 Cumulative periodogram, 22, 51, 172
 Cybenko's theorem, 207
- Data sets**
- Airline Data, 1, 2, 4, 8, 9, 12, 18, 20, 22, 23, 31, 35, 61, 115, 203, 204, 219, 582
 - Artificial data, 2, 4, 8, 9, 12, 20, 23, 31, 32, 292, 582
 - Buffalo Snowfall, 63, 589
 - Beveridge wheat price, 2, 6, 7, 8, 9, 12, 20, 23, 44, 45, 62, 580
 - Critical radio frequencies, 194, 195, 221, 222, 229, 248, 249, 294, 590
 - Daily California births, 2, 6, 7, 9, 12, 18, 20, 23, 579
 - Gas furnace data, 2, 3, 9, 333-335, 339-342, 344, 345, 585
 - Lake Erie levels, 2, 4, 9, 12, 20, 23, 583
 - Luteinizing hormone levels, 294, 591
 - Lynx data, 218, 219, 221, 227, 242, 253-255, 257, 260, 287, 288, 587
 - Magnitude of a star, 194, 195, 248, 249, 588
 - Mauna Loa CO₂ data, 589
 - Monthly sunspot data, 194, 195, 248, 249, 590
 - New York temperatures and births, 592
 - Normal white noise data, 2, 3, 4, 9, 12, 18, 20, 22, 23, 581
 - Rainfall data, 2, 4, 9, 12, 18, 20, 22, 23, 264, 265, 292, 581
 - Random walk series, 2, 8, 9, 11, 12, 18, 20, 23, 583
 - Sales data, 232-240, 245, 294, 587
 - Sum of four cosines, 2, 4, 9, 12, 18, 20, 23, 27, 565
 - Waves data, 290, 291, 593
 - Wolfer sunspot data, 69, 254, 255, 292, 293, 587
- Delay process, 317, 356
 Designing filters, 83
 Difference equations, 38, 61, 67, 201, 325
 Differencing, 30, 59, 60, 65, 79, 81, 234
 Dirichlet kernel, 82, 146, 157, 182
 Discrete Fourier transform (DFT), 15, 26
 Disturbed periodicity, 257
 DOS commands in TIMESLAB, 420, 434
 Dual process, 216
 Duality of ARMA processes, 117
- Editing files in TIMESLAB, 445-447
 Eigenvalues, 143
 Ensemble, 76
 Equivalent degrees of freedom, 162, 291, 296
 Ergodic, 162
 Errors in TIMESLAB, 428, 429
 Excess of high frequency, 18
 Excess of low frequency, 18
 Exponential decay, 104
 Exponential smoothing, 38, 57, 62, 66
 Extending data by inverse differencing, 60
- Factoring covariance generating functions, 134, 140, 204
 Fast Fourier transform (FFT), 24, 26, 371-374
 Fejér kernel, 169, 266
 Filter, AR, 129, 290
 Filter coefficients, 319
 Filter design, 83
 Filter, optimal, 318

- Filtered Poisson process, 254
- Filters, 78–86
- Filter theorem, 79, 314
- FIND mode in TIMESLAB, 419, 440–442
- Fisher's exact test, 256, 286, 299
- Forecasting, 35–42, 86–90, 118–134, 230–240
- Forward prediction coefficients, 123
- Four-fifths rule, 192
- Fourier coefficients, 181, 365
- Fourier pairs, 24
- Fourier series, 181, 363–371
- Fourth-order cumulant, 166
- Fourth-order stationary, 166
- Frequency components, 11, 27, 75, 81
- Frequency response function, 83
- Frequency transfer function, 80, 153, 354
- Fundamental frequency, 59
- Gain of a filter, 317
- Gain spectrum, 317, 356
- Gaussian time series, 70
- General linear model, 388
- General linear process, 117, 170
- Generalized least squares, 396
- Geometric series, 14, 26, 65
- Gibb's phenomenon, 84
- GPAC array, 216
- Gram-Schmidt decomposition, 63, 360
- Graphics in TIMESLAB, 435–445
- Hannan-Quinn modification of AIC, 222
- Harmonics, 19, 59
- Harmonic analysis, 11, 75
- Harmonic process, 92, 369
- Help in TIMESLAB, 430–431
- Hermitian matrix, 305
- High pass filter, 83
- Histograms, 42, 46, 62
- Homogeneous difference equation, 40, 67
- Horizon of a predictor, 86
- Hurwitz factor, 134
- Identifying ARMA models, 214–240
- Infinite memory prediction, 88
- Infinite order processes, 90, 101, 247
- Informative quantile function, 42, 46, 63, 67
- Initial conditions for difference equations, 39
- Innovations, 88, 90
- Innovation variance, 88, 272
- Intervention analysis, 243
- Inverse autocorrelation, 115
- Inverse autocovariance, 115, 216
- Inverse differencing, 35, 62
- Inverse discrete transform (IDFT), 15
- Inverse partial autocorrelations, 216
- Inverse spectral density, 115
- Invertibility, 95
- Kalman filter algorithm, 118, 131, 382
- Kernel density estimation, 43, 178
- Lag, 6
- Lag operator, 84
- Lag window, 181
- Lag window generator, 183, 292
- Lattice filter algorithm, 208
- Laurent expansion, 102
- Leakage, 179, 193
- Least squares estimation
 - For AR processes, 212
 - For regression models, 391
- Lebesgue-Stieltjes Integral, 73, 368
- Levinson's algorithm, 88, 205, 327
- Likelihood function, 119
- Limit in mean square, 78, 386
- Linear filters, 78–86
- Linear trend, 30
- Log transform, 30
- Long memory time series, 4, 36
- Low pass filter, 83

- MA spectral estimation, 253
- Macros in TIMESLAB, 447–465, 599–601
- Markov process, 158
- Markovian representation of ARMA process, 132
- Martingale process, 92
- Matrices in TIMESLAB, 422, 473
- Matrix inversion lemma, 363
- Maximum likelihood estimation, 120, 196–199
- Memory of a predictor, 86
- Method of moments estimators, 204
- Missing data, 52, 300
- Modified Cholesky decomposition, 96, 118–131, 358–360
- Monthly means and variances, 33, 62
- Moving average (MA) process, 78, 94–98
 - Algorithms, 134–140, 152
 - Bivariate, 314
 - Nonidentifiability, 96, 153
 - Estimation of, See ARMA Processes
 - Examples of, 98
 - Partial autocorrelations of, 97
 - Prediction of, 96, 131
 - Properties, 80, 94
 - Simulating, 86
- Moving average smoother, 56, 78, 81
- Multiple correlation, 10
- Multiplicative subset ARIMA process, 114, 230
- Multivariate normal distribution, 376
- Natural frequencies, 17, 170, 177
- Non-Gaussian time series, 118
- Nonidentifiability of MA processes, 96, 153
- Nonlinear time series, 118, 155
- Nonparametric probability density estimation, 43
- Nonparametric spectral density estimation, 176–194
- Optimal bandwidth, 45, 192
- Optimal filter, 318
- Orthogonal contrasts, 11
- Orthogonal regression, 397
- Orthogonality of sinusoids, 13
- Padding with zeros, 24, 27–29
- Parametric spectral density estimation, 246–254
- Parseval's relation, 366
- Parsimony, 230
- Partial autocorrelation function
 - Definition, 88
 - Estimation, 10, 210
 - For AR process, 106, 126
 - For ARMA process, 112
 - For MA process, 96
 - Inverse, 216
 - Relation with prediction, 89
- Parzen kernel, 67, 84, 184
- p -Differentiable, 188, 296
- Peak frequency estimation, 257, 294
- Periodic autoregressive process, 107, 357
- Periodogram, See Sample spectral density
- Permutation matrix, 87, 122, 154, 157
- Phase of a complex number, 13, 59, 63
- Phase spectrum, 307, 310, 335, 338
- Polynomials, 42, 125, 474
- Portmanteau test, 174, 268
- Positive definiteness, 72, 164, 358
- Power transformation, 30, 232
- Prediction, 35–42, 86–90, 118–134, 230–240
- Prediction normal equations, 88, 89, 123
- Prediction variance horizon (PVH), 218, 275, 294
- Prewhitening, 349
- Printing in TIMESLAB, 433, 439, 475

- Purely nondeterministic time series, 88
- Q test, 174, 268
- Quadrature spectral density, 307–310
- Quasi-differencing, 244
- Random walk process, 90–92, 102, 148, 153, 156, 291
- Recursive regression, 403
- Regression analysis, 31, 36, 56, 61, 388–406
 - Example, 399–402
 - With AR errors, 240–243
- Residual variances, 10
- Resolving peaks, 194
- Sample autocorrelation function, 7
 - Calculation of, 24, 28
 - Definition, 6
 - Examples, 8
 - For sinusoid, 65
 - Sampling properties, 166
- Sample partial autocorrelation function
 - Calculation of, 210
 - Definition, 10
 - Examples, 11
 - Sampling properties, 208
- Sample spectral density function, 11–21
 - Calculation of, 24
 - Definition, 17
 - Examples, 18
 - Sampling properties, 168
- Sample spectral distribution function, 21
- Saturation, 251
- Saving files in TIMESLAB, 431
- Saving graphics screens in TIMESLAB, 442
- Sawtooth function, 54, 59
- Scatterplot, 6, 47
- Schur matrix, 125, 206, 252
- Schur test for stationarity, 125
- Search for periodicities, 160, 256
- Seasonal means and variances, 33, 66
- Seeds for random number generators, 434
- Serial correlation, 6
- Sidelobes, 82, 183
- Simple exponential smoothing, 39
- Simulating Data
 - AR processes, 106, 128
 - ARMA processes, 155
 - MA processes, 86
 - Random Walk, 92
- Simultaneous confidence intervals, 379
- Sinusoid, 13, 65
- Sinusoidal decomposition, 15
- Smoothing, 33
- Spectral density function
 - Definition, 73
 - For AR process, 101
 - For ARMA process, 109
 - For MA process, 80
 - Interpretation of, 75–76
- Spectral distribution function, 172
- Spectral representations, 72–76, 367–371
- Spectral window, 181, 270
- Spectral window generator, 183
- Square root of a matrix, 358
- Squared coherence spectrum, 307, 310
- Standardizing spectra, 19
- Starting values, 39, 196
- State space model, 246
- Stationary increments, 73
- Stepwise AR modeling, 227–229
- Stepwise ARMA modeling, 224–227
- Stepwise regression, 397
- Stochastic difference equation, 99
- Stochastic integral, 73
- Stochastic process, 370
- Strictly stationary, 71, 155
- Subset ARMA process, 112
- Sunspot cycle, 293

- Superimposing plots in TIMESLAB, 462
- Sweep algorithm, 361
- Testing
 - Independence, 348
 - Location of polynomial zeros, 125
 - White noise, 172-176
 - Zero coherence, 351
- TIMESLAB, the program
 - Commands, Appendix C
 - Dictionary of commands, 477
 - Editor, 445-447
 - FIND, 440-442
 - Graphics menu, 438
 - Guided tour, 412-421
 - Hardware and software needed, 408
 - Installing, 409
 - Macros, 447-465, 599-601
 - Outline of Commands, 467
 - Scaling and labeling graphs, 443
 - Tic mark labels, 444
- Toeplitz matrix, 71, 72, 87, 123, 157
- Tradeoff of bias and variance, 178
- Transfer function
 - Estimation, 342
 - Model, 353
- Transformations, 29, 231
- Truncation point, 182, 193
- Unbiased residual variances, 215
- Uncorrelated increments, 370
- Unit circle, 42
- White noise process, 76, 311
- White noise tests, 50, 172-176
- Wilson's algorithms, 134-142
- Window spectral estimates, 176-193
- Wold decomposition, 89
- Yule-Walker equations, 101, 123, 204
- Yule-Walker estimators (YWE), 205-210, 273

Price \$31.80 • Total \$34.42

2006886



P9-BRR-058

